PERSONAL SOFTWARE

LA PRIMA RIVISTA EUROPEA DI SOFTWARE PER PERSONAL COMPUTER

Spedizione in abbonamento postale gruppo III/70

TI99/4A raccolta

Personal Software

Anni 1983 - 1986

Copia riservata agli abbonati

TEXAS TI 99/4A

Input e stampa estesa

Riceviamo dal sig. Filippo Cerulo di Vitulano (Benevento) queste brevi routine che permettono di superare alcune limitazioni del 99/4A.

Sono un assiduo lettore della vostra rivista, nonché un malato di bittomania cronica. Dopo molti sacrifici, da sei mesi sono in possesso di un TI 99/4A. Propongo dunque alla vostra attenzione un paio di routine di cui da un po' di tempo non riesco a fare a meno in nessun programma.

Il 99/4A è un ottimo calcolatore, ma, a meno di non comprare il modulo "Extended Basic", ha qualche piccolo limite, però largamente giustificato dal basso prezzo d'acquisto.

I problemi più grandi nascono dall'impossibilità di stampare messaggi e ricevere input se non sull'ultima riga del display, con conseguente scroll verso l'alto di tutto il video.

Confesso che tutto ciò all'inizio mi ha creato dei problemi soprattutto per l'impossibilità di stampare (e cancellare) frasi di commento a figure e grafici prodotti per mezzo dell'ottimo set di istruzioni del TI Basic.

Ma con un po' di buona volontà alla fine ho scritto le due brevi routine che vi propongo e che eliminano tutti i problemi.

Fate girare le due subroutine con il breve programma che accludo per capire bene come funzionano. Alla comparsa del simbolo di prompt (>) battete una qualsiasi stringa. In caso di errore basta premere FCTN3 (ERASE) e ribattere la stringa.

```
2000 REM SBR INPUT ESTESO
2010 K=0
2020 IMP$=""
2030 CALL HCHAR(RIN,CIN+K,62)
2040 CALL KEY(5,COD,ST)
2050 IF ST=0 THEN 2040
2060 IF COD 7 THEN 2090
2070 CALL HCHAR(RIN,CIN,32,K+1)
2080 GOTO 2010
2090 IF COD=13 THEN 2140
2100 IMP$=IMP$&CHR$(COD)
2110 CALL HCHAR(RIN,CIN+K,COD)
2120 K=K+1
2130 IF COD=32 THEN 2030 ELSE 2040
2140 RETURN
```

Listato 1. *Subroutine "input esteso".*

```
100 CALL CLEAR
110 READ R,C,S1$
120 GOSUB 1000
130 READ RIN,CIN
140 GOSUB 2000
150 PRINT" LA STRINGA E' ";IMP$
160 DATA 10,5,BATTI UNA STRINGA,13,5
170 END
```

Listato 2. *Programma principale.*

```
1000 REM STAMPA ESTESA
1010 N=LEN(S1$)
1020 FOR I=1 TO N
1030 S2$=SEG$(S1$,I,1)
1040 V1=ASC(S2$)
1050 CALL HCHAR(R,C-1+I,V1)
1060 NEXT I
1070 RETURN
```

Listato 3. *Subroutine "stampa estesa".*

Non mi dilungo nell'illustrare le singole istruzioni che per i possessori di TI 99/4A saranno abbastanza chiare. Dalla subroutine 1000, sacrificando la chiarezza, possono essere eliminate due istruzioni: i tempi di esecuzione delle due versioni sono però pressoché uguali. Preciso che i due apici nella riga 2020 non rappresentano uno spazio ma una stringa nulla. ■

Programmare grafici con il TI 99/4A

Anche con il Texas si possono tracciare grafici ad alta risoluzione

di *Roberto Del Giudice*

Il presente programma permette la rappresentazione grafica di una o più funzioni di una o due variabili (cioè del tipo: $F(x)$ e $F(x,y)$), ottenendo una buona risoluzione.

Il programma utilizza le funzioni e le istruzioni del Basic standard, risultando perciò piuttosto lento; la mancanza della funzione HEX(n), che permette di trasformare un numero dal formato intero in quello equivalente in formato esadecimale e delle funzioni AND, OR e NOT sono la causa primaria di tale inconveniente. Il programma potrebbe risultare ulteriormente più veloce se si potessero inserire più istruzioni per ogni linea di programma.

Descrizione del programma

110 - 120 Inizializzazione parametri.

130 - 160 Ciclo entro il quale viene definita la funzione da tracciare. I valori X ed Y da essa assunti dovranno essere compresi tra i seguenti estremi: $0<X\leq 96$ $0<Y\leq 80$; valori esterni a tali intervalli non vengono elaborati.

990 - 1050 Inizio della subroutine di costruzione della matrice A(N,NY) di transcodifica. Viene controllato che le coordinate del punto da plottare siano nell'intervallo consentito, altrimenti ritorna direttamente al programma principale.

1060 - 1080 Trasformazione delle coordinate X, Y in "coordinate carattere".

N = numero del carattere su cui operare; NX e NY = coordinate del pixel da settare all'interno del carattere N.esimo.

1090 - 1130 Operazioni che consentono di poter fare a meno delle funzioni AND e OR. In ogni vettore riga della matrice A(N,NY) si genera la maschera del relativo N.esimo carattere grafico, utilizzando una rappresentazione binaria in cui i pixel settati sono rappresentati come 1. Vedi esempio in fondo all'articolo.

1200 - 1460 Subroutine per la trasformazione della matrice A(N,NY) in N stringhe C$ nel formato richiesto dalla CALL CHAR. Si noti come le istruzioni dalla linea 1280 alla 1380 potrebbero essere sostituite, con un conseguente notevole risparmio di tempo macchina, dalla funzione HEX$(A$) che però non è disponibile in Basic standard.

1500 - 1530 Inizio della subroutine finale che costruisce il diagramma della funzione

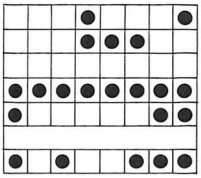
$F(x)$ o $F(x,y)$ sul video.

1540 - 1580 Viene "pulito" il video e quindi viene evidenziata l'area destinata al grafico tramite dei punti equidistanti.

1590 - 1610 I punti neri sono "trasformati" in punti bianchi.

1620 Viene cambiato il colore dello sfondo in nero.

NOTA Il passo di programma 180 ha lo scopo di mantenere il computer in elaborazione, evitando così l'immediata distruzione del grafico, una volta eseguito il programma.



A(6,1)=10001
A(6,2)=11100
A(6,3)=0
A(6,4)=11111111
A(6,5)=10000011
.....
A(6,8)=10100111

Esempio di codifica dei pixel di un carattere.

«PER ACCORCIARE I TEMPI»

il numero di TELEX

GRUPPO EDITORIALE JACKSON

è il seguente:

333436 GEJITI


```
 100 OPTION BASE 1
 110 DIM A(120,8)
 120 E$="10000000"
 130 FOR X=0 TO 96
 140 Y=(SIN(X/30)+1)*38          (Funzione di esempio)
 150 GOSUB 1020
 160 NEXT X
 170 GOSUB 1500
 180 GOTO 180
 990 REM SUBROUTINE COSTRUZIONE MATRICE DI TRANSCODIFICA
1000 REM
1010 REM
1020 IF Y>=80 THEN 1140
1030 IF Y<0 THEN 1140
1040 IF X>=96 THEN 1140
1050 IF X<0 THEN 1140
1060 N=INT(X/8)+(INT(Y/8)*12+1)
1070 NX=8-INT(X-INT(X/8)*8)
1080 NY=8-INT(Y-INT(Y/8)*8)
1090 A$=STR$(A(N,NY))
1095 IF (LEN(A$)-NX+1)<=0 THEN 1120
1100 IF SEG$(A$,LEN(A$)-NX+1,1)="1" THEN 1140
1110 REM
1120 D$=SEG$(E$,1,NX)
1130 A(N,NY)=A(N,NY)+VAL(D$)
1140 RETURN
1200 REM SUBROUTINE PER LA CREAZIONE DEI CARATTERI GRAFICI
1210 FOR K=1 TO 120
1220 C$=""
1230 FOR J=1 TO 8
1240 B$="00"
1250 IF A(K,J)=0 THEN 1420
1260 A$=STR$(A(K,J))
1270 H=0
1280 FOR Z=1 TO LEN (A$)
1290 H=H+VAL(SEG$(A$,Z,1))*2 ↑ (LEN(A$)-Z)
1300 NEXT Z
1310 B$=""
1320 A$="0123456789ABCDEF"
1330 FOR Z=INT(LOG(H)/LOG(16)) TO 0 STEP -1
1340 B=INT(H/16 ↑ Z)
1350 B$=B$&SEG$(A$,B+1,1)
1360 H=H-(16 ↑ Z)*B
1370 NEXT Z
1380 REM
1390 IF LEN(B$)<>1 THEN 1420
1410 B$="0"&B$
1420 C$=C$&B$
1430 NEXT J
1440 CALL CHAR(32+K,C$)
1450 NEXT K
1460 RETURN
1500 REM SUBROUTINE DI PRESENTAZIONE DEL GRAFICO
1505 CALL CLEAR
1510 FOR T=33 TO 153
1520 CALL CHAR(T,"18")
1530 NEXT T
1540 FOR I=1 TO 10
1550 FOR J=1 TO 12
1560 CALL VCHAR (17-I,10+J,20+J+I*12)
1570 NEXT J
1580 NEXT I
1590 FOR I=1 TO 16: CALL COLOR(I,16,1): NEXT I
1620 CALL SCREEN(2)
1630 GOSUB 1200
1640 RETURN
```

Archivio scacchistico per TI 99/4A

Con questo programma potete memorizzare su nastro le partite che più vi interessano, rivedendole a vostro piacere

di *Sergio Borsani*

Non molto tempo fa lessi in una rivista che in un questionario proposto ai candidati per un corso di programmazione, tra le altre domande, veniva chiesto se sapevano giocare a scacchi. Come è risaputo questo gioco presuppone in chi lo pratica capacità di analisi e di concentrarsi a lungo su un problema. Dalle premesse si può arguire che anch'io sono un giocatore di scacchi, forse al livello della terza categoria, anche se non ho mai partecipato ad un torneo di qualificazione. Da molto tempo raccolgo ritagli di riviste dove sono riportate partite di scacchi e le vado ordinando in un piccolo archivio. Dall'idea di computerizzare le informazioni raccolte è nato questo primo programma che, senza eccessive velleità, assolve in modo semplice e piacevole al suo compito.

Il programma è scritto in TI Basic e necessita del registratore come memoria di massa per la creazione di un file, cioè è richiesta la sola configurazione base.

Si distinguono due sezioni principali. Le considereremo separatamente.

La prima parte viene usata per memorizzare una partita. Selezionato il numero 1. appare la scritta REGISTRARE UNA PARTITA, e vengono chiesti in input una serie di dati: nome del giocatore con il bianco, nome del giocatore con il nero, luogo e data ed infine il tipo di apertura. Luogo e data formano un'unica stringa, pertanto non devono essere separati da una virgola. Tale segno serve, infatti, a separare le variabili in input. Al termine di ogni dato premete ENTER.

Tutti questi dati vanno a formare il primo record del file. Poiché nella OPEN non è stata specificata la lunghezza del record, il suo valore di default è di 64 caratteri e bisognerà stare attenti a non superare tale limite. La ragione per cui più variabili vengono riunite a formare un unico record è che si può sfruttare meglio la memoria di massa e rendere inferiori i tempi di registrazione e di caricamento. Se creiamo un record con un solo nome, ad es. Rossi, il computer completerà il record, con degli spazi in formato DISPLAY o con degli zeri binari in formato INTERNAL, fino a fargli raggiungere la lunghezza minima di 64 caratteri. Inoltre il computer, tra un record e l'altro, crea un gap, cioè uno spazio privo di registrazione; così, tra una sosta e l'altra, si dà modo al nastro di raggiungere l'adeguata velocità di registrazione. Insomma, se non fossero riuniti in modo ottimale, i dati nella memoria potrebbero occupare uno spazio anche 10 volte maggiore, allungando conseguentemente anche i tempi di utilizzazione.

I dati che costituiscono la notazione delle mosse hanno una lunghezza fissa di 12 caratteri, 6 per la mossa del bianco e 6 per il nero. Tale notazione, pur essendo abbastanza aderente a quella comunemente usata, presenta rispetto ad essa qualche differenza. Il primo carattere rappresenta il pezzo che viene mosso; questo dato non ha alcuna influenza nell'elaborazione tuttavia deve essere presente altrimenti si altera la lunghezza della stringa (uno spazio all'interno di una stringa è calcolato come un carattere mentre se è posto all'inizio no). Ad esempio, la mossa "e4" si deve trascrivere PE2E4−, indicando che un pedone si è spostato dalla casa e2 alla casa e4. Il numero di mossa appare automaticamente e quindi non deve essere trascritto. "2. Cf3 − Cg6" diventa:

CG1F3−CB8C6; Gl e B8 sono le rispettive case di partenza. La notazione deve essere esatta perché il programma non è in grado di rilevare eventuali errori, se scrivete RE1E4− ecc., durante la riproduzione della partita, il computer farà eseguire al Re un balzo ignorando completamente le regole degli scacchi. L'ultimo carattere di ogni semistringa, il sesto e il dodicesimo del record, normalmente non interviene nell'elaborazione, non può tuttavia essere una virgola, altrimenti compare il messaggio di errore e dovete riscrivere l'ultimo dato input. Questo carattere può essere utilizzato come commento; ad esempio, se c'è la presa di un pezzo, può essere ":", se c'è uno scacco al re, può essere il segno "+", oppure "!" se la mossa è buona e ancora "#" se c'è lo scacco matto. Normalmente si può usare il segno "−" o "=". Ci sono tuttavia alcuni casi in cui questo sesto carattere della semistringa influisce nell'elaborazione. Innanzi tutto se è ".", nella seconda semistringa, vogliamo indicare che la partita è finita; pertanto, quando si verifica questa eventualità e l'ultima mossa è del bianco, si completerà la stringa con degli spazi e si terminerà con il punto. Un altro caso particolare è la presa al passaggio; questa si segnala posponendo alle coordinate il simbolo "%". La promozione di un pedone esige che si specifichi il pezzo con il quale viene cambiato; D è la regina, T la torre, A l'alfiere e C il cavallo. Il simbolo va inserito come sesto carattere della semistringa. Inoltre si indicherà con −0−0− l'arrocco corto e con 0−0−0 l'arrocco lungo.

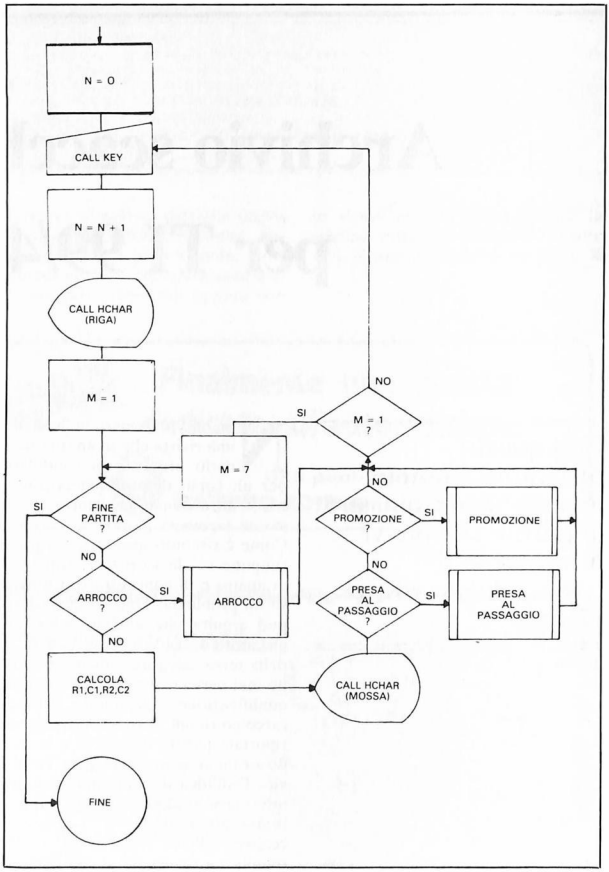


Figura 1. *Diagramma di flusso della parte riguardante la stampa. Il ciclo viene percorso due volte, la prima, con M=1, in cui viene letta la prima semistringa (mossa del bianco), la seconda, con M=7, in cui vengono letti gli ultimi sei caratteri (mossa del nero).*

Come vedete la notazione delle mosse è quasi uguale a quella convenzionale, la quale risulta soltanto più semplificata.

Le partite possono essere registrate su nastro una di seguito all'altra oppure a partire da numeri prefissati del contatore del vostro registratore.

Le partite trascritte su nastro

possono essere in seguito richiamate ed eseguite sullo schermo; questo avviene selezionando il numero 2. Quando appare la scritta REWIND CASSETTE TAPE CS1, dovete posizionare il nastro all'inizio di un file. Il caricamento avviene in un tempo che dipende dalla lunghezza della partita e generalmente dovrebbe durare uno o due minuti.

Caricata la sequenza di mosse, sul video appare una scacchiera, i nomi dei giocatori, il tipo di apertura, ecc. Ogni volta che viene premuto un tasto appare la notazione scritta della mossa e quindi questa viene eseguita sulla scacchiera. Per i dati scritti ho creato uno scroll verticale, come se occupassero una "finestra" e questo per non coinvolgere nello scrolling anche la scacchiera. Così appaiono scritte fino a dodici mosse; quando si passa alla tredicesima, scompare la prima. La possibilità di vedere, oltre ai movimenti sulla scacchiera, anche la notazione scritta è particolarmente utile quando si crea un archivio di aperture. Alla pressione di un tasto la partita prosegue fino a quando viene incorporato il punto al termine della stringa. A questo momento è possibile rivedere tutta la partita o passare ad un'altra.

Si consiglia di apportare alcune modifiche al programma per ottenere uno scrolling verso l'alto e verso il basso di tutte le mosse della partita e di programmare un tasto per poter riprendere l'esecuzione della partita dall'inizio senza dover attendere il suo completamento.

Descrizione del programma

Alla linea 210 viene dimensionata la matrice uni-dimensionale A$, ogni suo elemento conterrà una mossa. In 220-280 compare il menù, in base alla scelta si passa alle linee 340 per l'output, alla riga 790 per l'input del file e alla linea 2810 per uscire dal programma. Per i principianti aggiungo che si può uscire in ogni momento dal programma premendo FCTN(4).

Alla linea 350 iniziano le istruzioni INPUT per l'introduzione dei dati e le istruzioni relative. La linea 580 fa in modo che i numeri progressivi vengano incolonnati in corrispondenza delle unità. Alla linea 610 si rileva se la stringa in input ha lunghezza diversa da 12 caratteri e, a 640, se questa termina con un punto, si esce dal ciclo di input. Se si ritiene di aver trascritto correttamente i dati si può passare alla registrazione; l'OPEN viene data alla linea 700, su nastro il file non può che essere organizzato in modo sequenziale, l'apertura è in output poiché i dati devono passare ad un mezzo esterno, INTERNAL si riferisce al formato di registrazione, infine FIXED indica che tutti i record hanno la stessa lunghezza (64 byte). Come si può notare ogni record comprende 4 mosse. In 800-1190 si definiscono i caratteri ed i colori che formeranno la scacchiera. Le successive variabili con indice YY serviranno in seguito nella subroutine "promozione di un pedone". In 1250 si trova la OPEN in input per richiamare i dati in memoria centrale. Notate che i record vanno richiamati con riferimento alla stessa struttura usata per l'output, cioè, se il record è formato da quattro campi alfanumerici, anche se si usano nomi diversi, devono sempre essere richiamate quattro variabili di stringa. Interessanti sono i cicli delle linee 1450 e seguenti, il loro scopo è quello di creare il grafico della scacchiera senza ricorrere a 64 istruzioni CALL HCHAR, tante sono le case della scacchiera. Nella linea 1740 una CALL KEY dà il via all'esecuzione di una mossa. I REM inseriti nel listato vi permetteranno di comprendere l'ordine in cui vengono eseguite le varie operazioni. Il *clou* del programma è la parte adibita a riconoscere il pezzo presente nella casa di partenza, spostarlo nella casa d'arrivo e rimettere una casella vuota nella precedente.

Queste operazioni non sono così semplici come potrebbe apparire in un primo istante, infatti un pezzo bianco su sfondo scuro potrebbe spostarsi su una casella chiara ed in questo caso il pezzo non può essere definito dallo stesso carattere. In questo programma il Re bianco su fondo scuro è il carattere 128, mentre il RE bianco su fondo chiaro è il carattere 136. Alle linee 1960-1990 avviene la decodifica delle coordinate di ogni mossa che vengono trasformate in numeri di riga (R1 ed R2) e di colonna (C1 e C2) dello schermo. La seguente CALL GCHAR riconosce il pezzo della casa di partenza (PX). Si possono verificare quattro situazioni condizionate dai colori dello sfondo: da chiaro a chiaro, da chiaro a scuro, da scuro a chiaro ed infine da scuro a scuro. Il numero di carattere sarà invariato se resterà invariato lo sfondo (PX=PX) ma cambierà al variare di quello (PX=PX+8 oppure PX=PX−8). Il computer sa se una casa è chiara o scura sommando il numero di linea con il numero di colonna, se la somma è pari la casa è chiara, se è dispari è scura. In 2150-2240 c'è il controllo su un'eventuale presa al passaggio o sulla promozione di un pedone; le relative routine si trovano dopo la linea 2250 che chiude il ciclo.

VIDEO GIOCHI
LA PRIMA E UNICA RIVISTA DI VIDEOGAMES COMPUTER GIOCHI ELETTRONICI
Una pubblicazione del Gruppo Editoriale Jackson

Listato 1

```
100 REM  PERSONAL SOFTWARE
110 REM  ******************
120 REM  *                *
130 REM  *    ARCHIVIO    *
140 REM  *  SCACCHISTICO  *
150 REM  *                *
160 REM  ******************
170 REM  DI SERGIO BORSANI
180 REM        23.6.83
190 REM  TI-99/4A TI BASIC
200 CALL CLEAR
210 DIM A$(100)
220 PRINT TAB(11);"ARCHIVIO"
::::
230 PRINT TAB(9);"SCACCHISTI
CO":::::::
240 PRINT "PREMI:"::
250 PRINT "1.: REGISTRARE UN
A PARTITA"::
260 PRINT "2.: RICHIAMARE UN
A PARTITA"::
270 PRINT "3.: STOP":::::
280 INPUT "SCELTA? ":S$
290 IF S$="1" THEN 340
300 IF S$="2" THEN 790
310 IF S$="3" THEN 2810
320 GOTO 280
330 REM  *** SAVE ***
340 CALL CLEAR
350 PRINT "  REGISTRARE UNA
PARTITA":::::
360 PRINT "SCRIVI I NONI DEI
 GIOCATORI"
370 INPUT "BIANCO ":G1$
380 INPUT "NERO   ":G2$
390 PRINT
400 PRINT "LUOGO E DATA"
410 INPUT LD$
420 PRINT
430 INPUT "APERTURA:  ":AP$
440 PRINT
450 N=0
460 PRINT "SCRIVI LE MOSSE U
NA AD UNA"
470 PRINT "ES.: PE2E4-PC7C5-
":::
480 PRINT "PER FINIRE TERMIN
A CON UN  PUNTO"::
490 PRINT "AL PASSAGGIO: %":
:
500 PRINT "PROMOZIONE: D,T,A
, OPPURE C":::
510 PRINT "ARROCCO: -0-0- OP
PURE 0-0-0":::
520 PRINT TAB(9);"__________
_"
530 N=N+1
540 IF N<101 THEN 580
550 PRINT "* MEMORY FULL"
560 CALL SOUND(1000,440,2)
570 GOTO 200
580 PRINT TAB(6-LEN(STR$(N))
);STR$(N);". ";
590 INPUT A$(N)
600 CALL HCHAR(23,9,32)
610 IF LEN(A$(N))=12 THEN 64
0
620 PRINT "ERRORE! RIPROVA"
630 GOTO 580
640 IF SEG$(A$(N),12,1)="."
THEN 660
650 GOTO 530
660 CALL CLEAR
670 INPUT "VUOI REGISTRARE?
(S/N) ":RS$
680 IF SEG$(RS$,1,1)="S" THE
N 700
690 GOTO 340
700 OPEN #1:"CS1",SEQUENTIAL
,OUTPUT,INTERNAL,FIXED
710 K=1
720 PRINT #1:G1$;G2$;LD$;AP$
730 PRINT #1:A$(K);A$(K+1);A
$(K+2);A$(K+3)
740 K=K+4
750 IF K>N THEN 770
760 GOTO 730
770 CLOSE #1
780 GOTO 200
790 CALL CLEAR
800 RE$="081C087F493E1C00"
810 DA$="000049497F7F3E00"
820 AL$="0008183C3E087F00"
830 CA$="107C7E0E1E0C3E00"
840 TO$="002A3E1C1C1C3E00"
850 PE$="0000080808103E00"
860 CALL CHAR(128,RE$)
870 CALL CHAR(129,DA$)
880 CALL CHAR(130,AL$)
890 CALL CHAR(131,CA$)
900 CALL CHAR(132,TO$)
910 CALL CHAR(133,PE$)
920 CALL CHAR(134,"O")
930 CALL CHAR(136,RE$)
940 CALL CHAR(137,DA$)
950 CALL CHAR(138,AL$)
960 CALL CHAR(139,CA$)
970 CALL CHAR(140,TO$)
980 CALL CHAR(141,PE$)
990 CALL CHAR(144,RE$)
1000 CALL CHAR(145,DA$)
1010 CALL CHAR(146,AL$)
1020 CALL CHAR(147,CA$)
1030 CALL CHAR(148,TO$)
1040 CALL CHAR(149,PE$)
1050 CALL CHAR(152,RE$)
1060 CALL CHAR(153,DA$)
1070 CALL CHAR(154,AL$)
1080 CALL CHAR(155,CA$)
1090 CALL CHAR(156,TO$)
1100 CALL CHAR(157,PE$)
1110 CALL CHAR(158,"O")
1120 CALL CHAR(34,"000000000
00000FF")
1130 CALL CHAR(36,"010101010
1010101")
1140 CALL CHAR(64,"808080808
0808080")
1150 CALL CHAR(38,"FF")
1160 CALL COLOR(13,16,6)
1170 CALL COLOR(14,16,11)
1180 CALL COLOR(15,2,6)
1190 CALL COLOR(16,2,11)
1200 YY(1)=129
1210 YY(2)=132
1220 YY(3)=130
1230 YY(4)=131
1240 REM  *** OLD ***
1250 OPEN #1:"CS1",INPUT ,SE
QUENTIAL,INTERNAL,FIXED
1260 K=1
1270 INPUT #1:G1$,G2$,LD$,AP
$
1280 INPUT #1:A$(K),A$(K+1),
A$(K+2),A$(K+3)
1290 FOR J=K TO K+3
1300 IF SEG$(A$(J),12,1)="."
 THEN 1340
1310 NEXT J
1320 K=K+4
1330 GOTO 1280
```

(segue)

Segue Listato 1.

```
1340 CLOSE #1
1350 CALL CLEAR
1360 PRINT TAB(14-LEN(LD$)/2
);LD$;;;
1370 PRINT TAB(14-LEN(AP$)/2
);AP$;;;;;
1380 PRINT TAB(3);G2$;;;;;;;
;;;;
1390 PRINT TAB(3);G1$;;;
1400 PRINT "      * PREMI UN
TASTO *"
1410 CALL HCHAR(10,5,34,8)
1420 CALL VCHAR(11,4,36,8)
1430 CALL VCHAR(11,13,64,8)
1440 CALL HCHAR(19,5,38,8)
1450 DATA 156,147,154,145,15
2,146,155,148
1460 RESTORE 1450
1470 FOR J=1 TO 8
1480 READ PEZZO
1490 CALL HCHAR(11,4+J,PEZZO
)
1500 NEXT J
1510 FOR J=1 TO 8 STEP 2
1520 CALL HCHAR(12,4+J,149)
1530 CALL HCHAR(12,5+J,157)
1540 NEXT J
1550 Q=12
1560 FOR RIGA=13 TO 16
1570 FOR J=1 TO 8 STEP 2
1580 CALL HCHAR(RIGA,4+J,146
+Q)
1590 CALL HCHAR(RIGA,5+J,146
-Q)
1600 NEXT J
1610 Q=-Q
1620 NEXT RIGA
1630 FOR J=1 TO 8 STEP 2
1640 CALL HCHAR(17,4+J,141)
1650 CALL HCHAR(17,5+J,133)
1660 NEXT J
1670 DATA 132,139,130,137,12
8,138,131,140
1680 RESTORE 1670
1690 FOR J=1 TO 8
1700 READ PEZZO
1710 CALL HCHAR(18,4+J,PEZZO
)
1720 NEXT J
1730 N=0
1740 CALL KEY(0,KEY,STATUS)
1750 IF STATUS=0 THEN 1740
1760 N=N+1
1770 REM  *** STAMPA RIGA **
*
1780 W=N
1790 FOR RIGA=20 TO 9 STEP -
1
1800 RIG$=STR$(W)&" "&A$(W)
1810 IF W<1 THEN 1890
1820 FOR J=1 TO LEN(RIG$)
1830 L=ASC(SEG$(RIG$,J,1))
1840 CALL HCHAR(RIGA,14+J,L)
1850 NEXT J
1860 W=W-1
1870 NEXT RIGA
1880 REM  *** MOSSA ***
1890 M=1
1900 IF M=1 THEN 1920
1910 M=7
1920 MOSSA$=SEG$(A$(N),M,6)
1930 IF SEG$(MOSSA$,1,5)="-0
-0-" THEN 2270
1940 IF SEG$(MOSSA$,1,5)="0-
0-0" THEN 2380
1950 IF SEG$(MOSSA$,6,1)="."
 THEN 2760
1960 R1=19-VAL(SEG$(MOSSA$,3
,1))
1970 C1=ASC(SEG$(MOSSA$,2,1)
)-60
1980 R2=19-VAL(SEG$(MOSSA$,5
,1))
1990 C2=ASC(SEG$(MOSSA$,4,1)
)-60
2000 CALL GCHAR(R1,C1,PX)
2010 IF PX>M*3+130 THEN 2080
2020 VECCHIO=134
2030 IF (R2+C2)/2=INT((R2+C2
)/2)THEN 2060
2040 PEZZO=PX
2050 GOTO 2130
2060 PEZZO=PX+8
2070 GOTO 2130
2080 VECCHIO=158
2090 IF (R2+C2)/2=INT((R2+C2
)/2)THEN 2120
2100 PEZZO=PX-8
2110 GOTO 2130
2120 PEZZO=PX
2130 CALL HCHAR(R1,C1,VECCHI
O)
2140 CALL HCHAR(R2,C2,PEZZO)
2150 FF$=SEG$(MOSSA$,6,1)
2160 IF FF$="%" THEN 2500
2170 PRO=1
2180 IF FF$="D" THEN 2620
2190 PRO=2
2200 IF FF$="T" THEN 2620
2210 PRO=3
2220 IF FF$="A" THEN 2620
2230 PRO=4
2240 IF FF$="C" THEN 2620
2250 IF M=1 THEN 1910 ELSE 1
740
2260 REM  *** ARROCCO ***
2270 IF M=1 THEN 2330
2280 CALL HCHAR(11,12,134)
2290 CALL HCHAR(11,10,148)
2300 CALL HCHAR(11,9,158)
2310 CALL HCHAR(11,11,152)
2320 GOTO 2250
2330 CALL HCHAR(18,9,134)
2340 CALL HCHAR(18,11,128)
2350 CALL HCHAR(18,12,158)
2360 CALL HCHAR(18,10,140)
2370 GOTO 2250
2380 IF M=1 THEN 2440
2390 CALL HCHAR(11,9,158)
2400 CALL HCHAR(11,7,152)
2410 CALL HCHAR(11,5,158)
2420 CALL HCHAR(11,8,148)
2430 GOTO 2250
2440 CALL HCHAR(18,9,134)
2450 CALL HCHAR(18,7,128)
2460 CALL HCHAR(18,5,134)
2470 CALL HCHAR(18,8,140)
2480 GOTO 2250
2490 REM  *** AL PASSAGGIO *
**
2500 IF M=1 THEN 2560
2510 IF (R2+C2)/2=INT((R2+C2
)/2)THEN 2540
2520 CALL HCHAR(15,C2,158)
2530 GOTO 2250
2540 CALL HCHAR(15,C2,134)
2550 GOTO 2250
2560 IF (R2+C2)/2=INT((R2+C2
)/2)THEN 2590
2570 CALL HCHAR(14,C2,158)
2580 GOTO 2250
2590 CALL HCHAR(14,C2,134)
```

(segue)

Segue Listato 1.

```
2600 GOTO 2250
2610 REM  *** PROMOZIONE ***
2620 ORD=(R2+C2)/2
2630 IF ORD=INT(ORD)THEN 269
0
2640 IF M=1 THEN 2670
2650 PEZZO=YY(PRO)+16
2660 GOTO 2730
2670 PEZZO=YY(PRO)
2680 GOTO 2730
2690 IF M=1 THEN 2720
2700 PEZZO=YY(PRO)+24
2710 GOTO 2730
2720 PEZZO=YY(PRO)+8
2730 CALL HCHAR(R2,C2,PEZZO)
2740 GOTO 2250
2750 REM  *** FINE PARTITA *
**
2760 CALL HCHAR(23,1,32,64)
2770 PRINT "VUOI RIVEDERE LA
 STESSA"
2780 INPUT "PARTITA? (S/N) "
:RIS$
2790 IF SEG$(RIS$,1,1)="S" T
HEN 1350
2800 GOTO 200
2810 CALL CLEAR
2820 END
```

È vero: piccolo è bello!

Alla scoperta dello ZX SPECTRUM

a cura di **Rita Bonelli**

ZX Spectrum è l'ultimo nato della famiglia Sinclair. È un calcolatore a colori di piccole dimensioni, ma di grandissime possibilità. Imparare a usarlo bene può essere fonte di molte piacevoli scoperte. Questo libro vi aiuta a raggiungere lo scopo. In 35 brevi e facilissimi capitoli non solo imparerete tutto sulla programmazione in BASIC, ma arriverete anche a usare efficientemente il registratore e a sfruttare al meglio le stampe. Soprattutto capirete la differenza tra il vostro Spectrum e gli altri computer.

320 pagine. Lire 22.000 Codice 337 B

GRUPPO EDITORIALE JACKSON

Per ordinare il volume utilizzare l'apposito tagliando inserito in fondo alla rivista

TEXAS TI 99/4A

PRINT e INPUT in un contesto grafico

Sergio Borsani

In occasione del primo numero di questa rubrica dedicata ai possessori del TI 99/4A desidero rivolgere un particolare saluto tramite il breve programma di listato 1 che vi prego di mandare in esecuzione.

```
100 CALL CLEAR
110 DATA 67,73,65,79,32,84
120 DATA 69,88,65,78,73
130 RESTORE 110
140 FOR I=11 TO 21
150 READ L
160 CALL HCHAR(12,I,L)
170 NEXT I
180 CALL COLOR(2,16,7)
190 CALL HCHAR(10,9,42,15)
200 CALL VCHAR(11,9,42,3)
210 CALL VCHAR(11,23,42,3)
220 CALL HCHAR(14,9,42,15)
230 FOR I=1 TO 50
240 CALL COLOR(2,7,16)
250 CALL COLOR(2,16,7)
260 NEXT I
270 END
```

Listato 1.

La scritta è apparsa alla riga 12 e non nella parte bassa del monitor come avviene con l'istruzione PRINT.

La possibilità di accedere ad ogni posizione del video è particolarmente interessante quando si lavora in un ambiente grafico poiché l'esecuzione di ogni PRINT provoca lo spostamento verso l'alto di un disegno già presente. Per chi avesse ancora poca dimestichezza con il Basic, nel nostro esempio le istruzioni DATA contengono i codici ASCII delle lettere che formano la frase. L'istruzione RESTORE non è qui indispensabile, ma lo diventa nel caso di ripetute analoghe istruzioni in un programma articolato. Gli elementi contenuti nei DATA, vengono letti sequenzialmente indipendentemente dalla loro posizione, il RESTORE serve a riportare il puntatore nella posizione desiderata.

Lo stesso risultato può essere ottenuto utilizzando

```
110 A$="CIAO TEXANI"
120 FOR I=1 TO 11
130 L=ASC(SEG$(A$,I,1))
140 CALL HCHAR(12,10+I,L)
150 NEXT I
```

Listato 2.

funzioni di stringa. Le righe dalla 110 alla 170 possono essere sostituite da quelle riportate nel listato 2 e in questo modo si evita la conversione delle lettere in codice ASCII.

Il TI Extended Basic mette a disposizione il comando DISPLAY AT(x,y) che svolge le stesse funzioni consentendo un notevole risparmio di tempo e di memoria. Nel primo caso sono stati occupati 129 byte, nel secondo 98, con l'istruzione in Extended Basic solo 33.

In molte applicazioni si rende necessaria anche l'introduzione di dati in ogni posizione video. Anche in questo caso l'Extended Basic offre il comando ACCEPT AT(x,y) ma anche con il Basic residente nella consolle è possibile ottenere un effetto analogo usando l'istruzione CALL KEY. Consideriamo dapprima i suoi elementi di sintassi: CALL KEY(0,K,S). 0 definisce la tastiera, in questo modo il computer associa ad ogni tasto il valore ASCII corrispondente; tale valore viene attribuito alla variabile K; S assume invece i seguenti valori: 0 quando non viene premuto alcun tasto, −1 quando viene premuto lo stesso tasto premuto in precedenza, 1 quando viene premuto un nuovo tasto.

Un uso dell'istruzione CALL KEY, oltre a quello descritto nel manuale è riportato nel listato 3.

```
100 CALL CLEAR
110 C=10
120 CALL KEY(0,K,S)
130 IF S=0 THEN 120
140 A$=A$&CHR$(K)
150 C=C+1
160 CALL HCHAR(12,C,K)
170 GOTO 120
```

Listato 3.

Questi passi di programma corrispondono ad un INPUT nella riga 12 a partire dalla colonna 10, se ora digitate una frase, essa viene visualizzata in quella posizione e viene memorizzata come variabile di stringa A$. Infatti, se interrompete il ciclo premendo FCTN(4)=CLEAR e impostate il comando immediato (senza numero di linea) PRINT A$, ritorna la frase digitata in precedenza.

A conclusione voglio presentare un gioco che ben sintetizza tutte le considerazioni esposte (listato 4).

Si tratta di colpire un castello a colpi di cannone: vengono messi a disposizione 5 tiri e di volta in volta bisogna specificare l'alzo scrivendo i gradi. Il programma prevede due cifre per gli interi ed una per il valore decimale, pertanto, per scrivere un angolo di 8°, è necessario digitare 08 0.

Listato 4. *Il programma "Tiri balistici".*

```
100 REM  *** TIRI BALISTICI***
110 CALL CLEAR
120 PRINT " *************************"
130 PRINT " *                       *"
140 PRINT " *      TIRI BALISTICI     *"
150 PRINT " *                       *"
160 PRINT " *************************"
170 PRINT :::::
180 CALL CHAR(128,"6061F1F0F061667C")
190 CALL CHAR(129,"00FFFF70F8DBF870")
200 CALL CHAR(130,"0000000008142A55")
210 CALL CHAR(131,"000008 1C3E7F1C1C")
220 CALL CHAR(136,"00000000AAFFEE6E")
230 CALL CHAR(137,"AAAAFEFEFEFEEEEC")
240 CALL CHAR(138,"7F7F7F7171717171")
250 CALL CHAR(139,"FCFCFCFCFCFCFCFC")
260 CALL CHAR(144,"0000000000000001")
270 CALL CHAR(145,"00000010387CFEFF")
280 CALL CHAR(140,"000000040E1F7FFF")
290 CALL CHAR(141,"04204290B8DBCE46")
300 CALL CHAR(142,"030F1F1131717171")
310 CALL CHAR(143,"000104000A000501")
320 CALL CHAR(152,"FFFFFFFFFFFFFFFF")
330 CALL COLOR(16,3,3)
340 CALL COLOR(15,9,1)
350 CALL COLOR(14,16,1)
360 RANDOMIZE
370 D=INT((-3000+1)*RND)+5000
380 D=INT(D/10)*10
390 PRINT "DEVI COLPIRE UN BERSAGLIO A"
400 PRINT D;"METRI DI DISTANZA"
410 PRINT " ";"HAI A DISPOSIZIONE 5 TIRI"
420 PRINT " ";" ";" "
430 PRINT "PREMI 1. PER CONTINUARE"
440 CALL KEY(0,KEY,STATUS)
450 IF STATUS=0 THEN 440
460 CALL CLEAR
470 CALL HCHAR(20,1,152,160)
480 CALL HCHAR(19,4,130)
490 CALL HCHAR(19,5,128)
500 CALL HCHAR(19,6,129)
510 CALL HCHAR(17,24,144)
520 CALL HCHAR(17,25,145)
530 CALL HCHAR(18,24,136)
540 CALL HCHAR(18,25,137)
550 CALL HCHAR(19,24,138)
560 CALL HCHAR(19,25,139)
570 A$="ALZO? (MAX 90)    ERRORE M."
580 FOR I=1 TO LEN(A$)
590 A=ASC(SEG$(A$,I,1))
600 CALL HCHAR(1,2+I,A)
610 NEXT I
620 L=1
630 CALL HCHAR(2+L,3,48+L)
640 CALL HCHAR(2+L,9,44)
650 CALL HCHAR(3+L,7,131)
660 GRAD=0
670 FOR J=1 TO 3
680 IF J=3 THEN 710
690 W=J
700 GOTO 720
710 W=4
720 CALL KEY(0,KEY,STATUS)
730 IF STATUS=0 THEN 720
740 CALL HCHAR(2+L,6+W,KEY)
750 GRAD=GRAD+(KEY-48)*10^(2-J)
760 CALL HCHAR(3+L,6+W,32)
770 CALL HCHAR(3+L,7+J,131)
780 IF J=2 THEN 810
790 IF J=3 THEN 840
800 GOTO 850
810 CALL HCHAR(3+L,9,32)
820 CALL HCHAR(3+L,10,131)
830 GOTO 850
840 CALL HCHAR(3+L,10,32)
850 NEXT J
860 S$="PREMI 1. PER SPARARE"
870 FOR I=1 TO LEN(S$)
880 S=ASC(SEG$(S$,I,1))
890 CALL HCHAR(10,3+I,S)
900 NEXT I
910 CALL KEY(0,KEY,STATUS)
920 IF STATUS=0 THEN 910
930 CALL SOUND(300,-6,2)
940 CALL HCHAR(10,3,32,28)
950 RADI=GRAD*3.14159/180
960 X=100000*SIN(2*RADI)/9.8
970 ER=INT((X-D)/10)*10
980 IF ABS(ER)<=20 THEN 1000
990 GOSUB 1340
1000 CALL HCHAR(19,24,42)
1010 CALL HCHAR(18,24,141)
1020 CALL HCHAR(18,23,143)
1030 CALL SOUND(400,-6,1)
1040 CALL SOUND(300,-7,2)
1050 CALL SOUND(200,-6,3)
1060 CALL SOUND(400,-7,5)
1070 CALL SOUND(800,-7,10)
1080 CALL SOUND(1200,-7,15)
1090 CALL SOUND(1600,-7,20)
1100 CALL SOUND(2000,-7,25)
1110 CALL HCHAR(19,23,140)
1120 CALL HCHAR(19,24,142)
1130 CALL HCHAR(18,23,32,2)
1140 C$="COLPITO"
1150 FOR I=1 TO LEN(C$)
1160 C=ASC(SEG$(C$,I,1))
1170 CALL HCHAR(2+L,19+I,C)
1180 NEXT I
1190 RI$="VUOI RICOMINCIARE? (S/N)"
1200 FOR I=1 TO LEN(RI$)
1210 RI=ASC(SEG$(RI$,I,1))
1220 CALL HCHAR(10,3+I,RI)
1230 NEXT I
1240 CALL KEY(0,KEY,STATUS)
1250 IF STATUS=0 THEN 1240
1260 IF KEY=83 THEN 1280
1270 IF KEY<>115 THEN 1330
1280 CALL HCHAR(3,1,32,160)
1290 CALL HCHAR(10,1,32,30)
1300 CALL HCHAR(12,18,32,12
1310 CALL CLEAR
1320 GOTO 360
1330 END
1340 COEFF=D/18
1350 COL=INT(7+(X/COEFF))
1360 IF COL<24 THEN 1430
1370 IF COL>30 THEN 1480
1380 IF COL>25 THEN 1430
1390 IF COL=24 THEN 1420
1400 COL=26
1410 GOTO 1430
1420 COL=23
1430 CALL HCHAR(19,COL,42)
1440 CALL SOUND(300,-5,1)
1450 CALL SOUND(900,-6,10)
1460 CALL SOUND(1500,-7,20)
1470 GOTO 1530
1480 T$="TIRO LUNGO"
1490 FOR I=1 TO LEN(T$)
1500 T=ASC(SEG$(T$,I,1))
1510 CALL HCHAR(12,18+I,T)
1520 NEXT I
1530 ER$=STR$(ER)
1540 FOR I=1 TO LEN(ER$)
1550 E=ASC(SEG$(ER$,I,1))
1560 CALL HCHAR(2+L,25-LEN(ER$)+I,E)
1570 NEXT I
1580 L=L+1
1590 IF L>5 THEN 1620
1600 CALL HCHAR(12,16,32,12)
1610 GOTO 630
1620 RIS$="VUOI RICOMINCIARE? (S/N)"
1630 FOR I=1 TO LEN(RIS$)
1640 R=ASC(SEG$(RIS$,I,1))
1650 CALL HCHAR(1*0,3+I,R)
1660 NEXT I
1670 CALL KEY(0,KEY,STATUS)
1680 IF STATUS=0 THEN 1670
1690 IF KEY=83 THEN 1710
1700 IF KEY<>115 THEN 1760
1710 CALL HCHAR(3,1,32,160)
1720 CALL HCHAR(10,3,32,28)
1730 CALL HCHAR(12,18,32,12)
1740 CALL CLEAR
1750 GOTO 360
1760 END
```

Il taglio del TI99/4A

Un avvincente gioco di strategia

di *Alessandro Chessa*

Non molto tempo fa, nel numero di Dicembre dell'anno scorso della rivista scientifica "Le Scienze", appariva, nella consueta rubrica "Temi Matematici" condotta brillantemente da Douglas R. Hofstadter, un interessante articolo intitolato "Taglio, Sfoggio, Hruska, evoluzione del comportamento e altri giochi di strategia", nel quale venivano presentati alcuni originali giochi numerici a carattere strategico-psicologico.

Tra i vari giochi, del resto tutti molto accattivanti, uno in particolare aveva stimolato la mia curiosità mentale. Intendo riferirmi al "Taglio", un gioco ideato dallo stesso autore e da un suo collega, Robert Boeninger. Le regole del gioco sono a dir poco semplici. Si tratta di scegliere mentalmente, ad ogni turno, due numeri, uno per ciascun giocatore (si gioca in due), "pescandoli" da uno stesso insieme numerico (numeri interi da 1 a 5) per poi confrontarli tra di loro; se la loro differenza risulta diversa da 1, ciascun giocatore terrà il proprio numero sommandolo al proprio punteggio; se invece la differenza è esattamente 1, il giocatore con il più basso dei numeri li sommerà entrambi, "tagliando" l'avversario che rimarrà all'asciutto. Quest'ultimo espediente è stato naturalmente escogitato per evitare che si possano scegliere troppo facilmente numeri grandi, rendendo così il gioco più movimentato e più articolato nel suo svolgimento.

Quello che più mi affascinava era la straordinaria possibilità, troppo spesso disattesa da sofisticati (come grafica) ma pur sempre limitati giochi spaziali, di prestarsi ad una infinità di soluzioni e di creare una partita che accumulasse livelli su livelli di gioco, limitati solo dalla fantasia e dalla disponibilità di tempo dei giocatori.

La bravura in questo gioco consiste nell'attivare una strategia fatta di astuzie psicologiche appositamente congeniate per sviare e poi "tagliare" l'avversario. Ciò si può ottenere per esempio, ostentando un determinato schema numerico, quale 4-4-4, per spingere l'avversario a dire 3 proprio nel momento in cui voi direte 2 (non sempre capita!).

È un gioco fatto di sottili tensioni psicologiche che spesso si risolvono in una serrata battaglia che ha come unico superstite (vincitore) il più saldo e forte di nervi.

Essendo possessore di un computer (TI99/4A) intravvedevo la possibilità, del resto già prospettata dallo stesso Hofstadter, di fare del mio computer un valente giocatore di Taglio.

Problemi e idee per un programma "esperto" di Taglio

Le maggiori difficoltà incontrate nell'ideare un programma "esperto" nel gioco del Taglio derivavano dal

fatto che non trattandosi di un gioco "deterministico" o quasi come ad esempio gli scacchi, non ci si poteva avvalere di tecniche euristiche, tipiche delle applicazioni nel campo dell'intelligenza artificiale. Per il taglio si deve più propriamente parlare di gioco "probabilistico", nel senso che la strategia di gioco si può fondare solo su previsioni, più o meno probabili, ricavate attraverso una ricerca statistica delle giocate precedenti che riveli delle ricorsività di schemi numerici, dove per schemi numerici intendo sequenze ben determinate di numeri. Occorreva, quindi, che il computer cogliesse gli schemi, tal volta anche involontari (e questo l'ho sperimentato personalmente), dell'avversario e, giocando in effetti sulla sua psicologia, se ne servisse per generare previsioni ragionevolmente fondate; ma questo non era evidentemente sufficiente. A questo riguardo vorrei fare un esempio abbastanza significativo. Se nel corso di una partita, per due giocate consecutive scegliamo il numero 1, il computer alla terza avendo individuato lo schema numerico 1-1, si aspetterà di nuovo un 1 e sarà quindi portato a giocare un 5, dato che in questa situazione è la scelta più vantaggiosa (5−1= +4). Naturalmente conoscendo già in precedenza la sua scelta, lo potremo facilmente tagliare con un 4, guadagnando ben 9 punti. Ora, pur ammettendo che in tutti e tre i turni il computer abbia giocato un 5, la somma totale dei punteggi risulta comunque a nostro favore. È chiaro che se provassimo a ripetere il tranello non incorrerebbe nello stesso errore, ma per evitare che lo stesso si possa verificare per altri schemi numerici, diversi da 1-1, si rende evidentemente necessaria una vera e propria strategia di gioco che diminuisca la prevedibilità delle mosse del computer.

Il mio programma cerca di realizzare tutto ciò sopra riferito.

Funzionamento del programma

Adesso in breve tenterò di spiegare come effettivamente funziona.

Una volta inserito il numero del giocatore (N) appare la scritta

```
 1 REM **********************
 2 REM **********************
 3 REM ****** TAGLIO *******
 4 REM **********************
 5 REM **********************
 6 REM **********************
 7 REM * PERSONAL SOFTWARE *
 8 REM **********************
 9 REM **********************
10 REM
12 REM
14 REM
16 REM
20 REM
21 REM ----------------------
22 REM   Gioco Strategico
24 REM   Ideato da:
26 REM Douglas R.Hofstadter
28 REM tratto da"Le Scienze"
30 REM ----------------------
32 REM **********************
34 REM ----------------------
36 REM Versione Computer di:
38 REM Alessandro Cheasa
40 REM Su Home Computer:              TI 99/4A(Texas I.)
41 REM Versione Basic:
42 REM *EXTENDED BASIC*
44 REM
46 REM Cagliari 17/4/1983
47 REM ----------------------
48 REM **********************
50 REM   Bibliografia:
52 REM   Le Scienze N.172
54 REM   Dicembre/82
56 REM **********************
58 REM
59 REM
60 REM ********************
61 REM *****VARIABILI******
62 REM ********************
63 REM
64 REM X(200)...VETT.PRINC.
65 REM Y(200)...VETT.SECON.
66 REM N.....NUM.GIOCATORE
67 REM C...CONTATTORE VETT.
68 REM NC...NUM.COMPUTER
69 REM NA...NUM.AVVERSARIO
70 REM CAS1  *
71 REM CAS2.P**NUM.CASUALI
72 REM CAS3  *
73 REM C1*
74 REM C2**
75 REM C3***CONTATTORI
76 REM C4***SUBROUTINES
77 REM C5**
78 REM C6*
79 REM TOT..PUNTEGG.GIOCAT.
80 REM CTOT.PUNTEGG.COMPUT.
81 REM W...C1,C2,C3,C4,C5
82 REM A$...VALIDATE: "SsNn"
83 REM DIF...NC-N
84 REM T.....NUMERO TURNI
85 REM ********************
86 REM ******COSTANTI*****
87 REM ********************
89 REM Z....1,2,3,4,5
90 REM ********************
91 REM **DIMENSIONAMENTO**
92 REM ********************
93 OPTION BASE 1
94 DIM X(200)
95 DIM Y(200)
96 REM
97 REM ********************
98 REM *INIZIALIZZAZIONE*
99 REM ********************
100 REM
105 LET C=0
110 LET NC=0
115 LET NA=0
120 LET C6=0
125 LET TOT=0
130 LET CTOT=0
132 REM
180 CALL CHAR(130,"3333CCCC3333CCCC")
182 CALL COLOR(13,6,8)
185 ON WARNING NEXT
190 RANDOMIZE
195 CALL SCREEN(10)
199 CALL CLEAR
200 PRINT "******************************"
203 PRINT "******************************"
205 PRINT "******************************"
210 PRINT "******************************"
213 PRINT "******************************"
215 PRINT "******************************"
220 PRINT "******************************"
225 PRINT "**********TAGLIO**********"
230 PRINT "******************************"
233 PRINT "******************************"
235 PRINT "******************************"
240 PRINT "******************************"
243 PRINT "******************************"
245 PRINT "******************************"
248 PRINT "******************************"
250 PRINT
251 CALL DELAY(2)
253 RESTORE 9500
255 READ A,B
257 IF A=-1 THEN 268
259 DISPLAY AT(A,B)SIZE(1)," "
261 CALL SOUND(-120,40000,30,40000,30,330,30,-8,1)
265 GOTO 255
268 CALL DELAY(3)
270 CALL SCREEN(8)
```

Listato 1. *Il programma BASIC.*

Seguito listato 1.

```
280 DISPLAY AT(10,7)ERASE ALL "VUOI CONOSCERE "
290 DISPLAY AT(12,3) "LE REGOLE DEL *TAGLIO*"
300 DISPLAY AT(15,9) "(S/N)?"
310 ACCEPT AT(15,16)BEEP SIZE(1)VALIDATE("SaNn") A$
320 IF A$="S" OR A$="s" THEN GOSUB 9000 :: CALL SCREEN(8)
322 DISPLAY AT(10,3)ERASE ALL "QUANTI TURNI VUOI GIOCARE"
324 DISPLAY AT(15,6) "(DA 50 A 200)?"
326 ACCEPT AT(15,22)BEEP VALIDATE(DIGIT)SIZE(-3) T
328 IF T<50 OR T>200 OR INT(T)<>T THEN 326
338 CALL SCREEN(8)
339 CALL CLEAR
340 DISPLAY AT(1,3) "PLAYER    TURN  COMPUTER"
345 IF T=0 THEN 650
350 DISPLAY AT(2,3) "SCORE      ";C :: DISPLAY AT(2,18) "SCORE"
360 DISPLAY AT(3,4)SIZE(4) TOT :: DISPLAY AT(3,19)SIZE(4) CTOT
365 CALL HCHAR(5,1,130,32)
370 DISPLAY AT(9,5) "PLAYER       COMPUTER"
380 DISPLAY AT(10,5) "NUMBER       NUMBER"
385 DISPLAY AT(13,7)SIZE(4) "+          ="
387 CALL HCHAR(19,1,130,32)
390 ACCEPT AT(13,7)BEEP VALIDATE(DIGIT)SIZE(-1) N
395 IF N<1 OR N>5 OR INT(N)<>N THEN 385
400 DISPLAY AT(16,17)BEEP "COMPUTER" :: DISPLAY AT(17,17) "THINKING"
500 GOSUB 1000
510 LET C&=0
520 DISPLAY AT(13,19)BEEP NC
530 LET DIF=NC-N
540 IF DIF=1 THEN TOT=TOT+N+NC :: CALL SCREEN(14) :: DISPLAY AT(21,17) "TAGLIATO'
" :: CALL DELAY(3) :: DISPLAY AT(21,1) "" :: GOTO 580
550 IF DIF=-1 THEN CTOT=CTOT+NC+N :: CALL SCREEN(14) :: DISPLAY AT(21,5) "TAGLIAT
O!" :: CALL DELAY(3) :: DISPLAY AT(21,1) "" :: GOTO 580
560 LET TOT=TOT+N
570 LET CTOT=CTOT+NC
580 CALL SCREEN(8)
590 CALL DELAY(3)
600 CALL HCHAR(16,1,32,64) :: GOTO 345
650 REM ********************
655 REM **MESSAGGI FINALI**
660 REM ********************
663 CALL SCREEN(12)
665 DISPLAY AT(2,7)BEEP ERASE ALL "PUNTEGGIO FINALE"
670 DISPLAY AT(5,4) "PLAYER        COMPUTER"
673 DISPLAY AT(6,5) TOT :: DISPLAY AT(6,19) CTOT
675 IF CTOT>TOT THEN DISPLAY AT(15,4)" EBBENESI' SONO ANCORA IO  ER MEJO DE
TUTTI."
680 IF CTOT=TOT THEN DISPLAY AT(15,9)"  PER QUESTA VOLTA TI E'  ANDATA BE
NE          E' FINITA IN PARITA'."
685 IF CTOT<TOT THEN DISPLAY AT(15,4)"  BRAVO!!! SEI RIUSCITO A  BATTERE UN
 COMPUTER."
690 CALL DELAY(20)
700 DISPLAY AT(8,5)ERASE ALL "VOUI INIZIARE UNA"
710 DISPLAY AT(10,5) "NUOVA PARTITA"
720 DISPLAY AT(15,7)SIZE(6) "(S/N)?"
730 ACCEPT AT(15,16)BEEP SIZE(1)VALIDATE("SNsn") A$
750 IF A$="S" OR A$="s" THEN 90
760 DISPLAY AT(16,9)ERASE ALL "ARRIVEDERCI!!"
850 STOP
900 REM ********************
902 REM ********************
903 REM ****SUBROUTINES****
905 REM ********************
906 REM ********************
907 REM
909 REM
910 REM
950 REM ********************
960 REM   SUBROUTINE 1000
970 REM (RICERCA STRATEGIA)
980 REM ********************
990 REM
999 REM
1000 LET C=C+1
1010 LET X(C)=N
1012 LET CAS1=INT(RND*5)+1
1015 LET CAS2=INT(RND*5)+1
1017 IF C>9 THEN 1070
1018 REM
1019 REM ********************
1020 REM STRATEGIA INIZIALE
1022 REM ********************
1023 REM
1024 IF C=1 THEN NC=CAS2 :: RETURN
1026 IF C=2 THEN IF CAS1=1 AND X(C-1)<>1 THEN NC=X(C-1)-1 :: RETURN ELSE NC=CAS2
   RETURN
1030 IF C=3 THEN IF X(C-1)=X(C-2)AND CAS1>1 THEN NA=X(C-1) :: GOTO 1080 ELSE NC=C
AS2 :: RETURN
1050 ON C-3 GOSUB 5500,5000,5000,5000,5000,5000
1060 GOTO 1080
1070 GOSUB 2000
1073 REM ********************
1075 REM      STRATEGIA
1077 REM ********************
1080 LET P=RND
1100 IF NA=0 AND T-C<=13 THEN IF CTOT-TOT<=-20 THEN NC=INT(RND*3)+3 :: RETURN
1110 IF NA=0 AND T-C<=13 THEN IF CTOT-TOT>=20 THEN NC=INT(RND*3)+1 :: RETURN
1120 IF NA=0 THEN NC=INT(RND*5)+1 :: RETURN
1140 IF NA=1 AND CTOT-TOT<=-20 THEN IF P<.70 THEN NC=5 :: RETURN ELSE NC=4 :: RE
TURN
1150 IF NA=1 AND CTOT-TOT>=20 AND P<.75 THEN IF P<.40 THEN NC=5 :: RETURN ELSE N
C=4 :: RETURN
1160 IF NA=1 AND CTOT-TOT>=20 THEN NC=3 :: RETURN
1170 IF NA=1 THEN IF P<.60 THEN NC=5 :: RETURN ELSE NC=4 :: RETURN
1180 IF NA=2 AND CTOT-TOT<=-20 THEN IF P<.70 THEN NC=1 :: RETURN ELSE NC=5 :: RE
TURN
```

"Computer Thinking" e il computer inizia la sua ricerca ignorando naturalmente il numero appena inserito. Dapprima considera gli ultimi 5 numeri scelti e va a ricercare se precedentemente sia stato già giocato quello schema. Ogni qualvolta ne incontra uno identico inserisce in un contenitore (il vettore Y) il numero subito seguente lo schema. Se alla fine della ricerca non ha trovato nessuno schema da 5 numeri passa a considerare schemi da 4 poi da 3, da 2 ed infine da 1 numero. Ammettiamo, per semplicità, che abbia individuato 10 schemi da 3 numeri e che i numeri seguenti questi schemi, ovvero i numeri di preferenza in quelle particolari circostanze, siano stati due 2 e otto 5. A questo punto il computer sceglie dal contenitore un numero (NA = probabile numero avversario) fra i dieci prescelti. Bisogna però considerare che la casualità è fortemente condizionata dalle proporzioni in cui sono stati giocati gli schemi. In questo caso particolare la scelta è ristretta ai numeri 2 e 5 e la probabilità che esca il 5 è di 8 parti su 10 (80% contro il 20% del 2). Una volta estratto il numero dal contenitore si procede all'applicazione della strategia. La strategia da me ideata fa uso di determinati *pesi*, espressi in percentuali, che sono assegnati alle diverse scelte possibili (tabella 2). Le diverse scelte sono a loro volta determinate dal numero estratto dal contenitore. I criteri adottati per la "calibrazione" dei pesi sono del tutto soggettivi, e sono quindi passibili di modifiche a seconda delle diverse interpretazioni tattiche. In generale, per quanto riguarda il criterio da me seguito, posso dire di aver tenuto conto di tre fattori fondamentali: il guadagno nel punteggio, la "tagliabilità" del numero e le condizioni di gioco contingenti. Mi spiegherò meglio con un esempio. Se il numero previsto dal computer (NA) è un 2 sarebbe indifferente per lui giocare un 1 od un 5 perché entrambi procurerebbero un più 3 nel conteggio del turno (infatti 1 + 2 = 5 − 2). Non sempre però il numero previsto è quello effettivamente scelto dall'avversario. Nell'incertezza l'1 è da preferirsi in quanto non può essere tagliato

Seguito listato 1.

```
1190 IF NA=2 AND CTOT-TOT>=20 AND P< 80 THEN IF P< 50 THEN NC=1 :: RETURN ELSE N
C=5 :: RETURN
1200 IF NA=2 AND CTOT-TOT>=20 THEN NC=4 :: RETURN
1210 IF NA=2 THEN IF P< 65 THEN NC=1 :: RETURN ELSE NC=5 :: RETURN
1220 IF NA=3 AND CTOT-TOT<=-20 THEN IF P< 80 THEN NC=2 :: RETURN ELSE NC=5 :: RE
TURN
1230 IF NA=3 AND CTOT-TOT>=20 AND P< 85 THEN IF P< 55 THEN NC=2 :: RETURN ELSE N
C=5 :: RETURN
1240 IF NA=3 AND CTOT-TOT>=20 THEN NC=3 :: RETURN
1250 IF NA=3 THEN IF P< 75 THEN NC=2 :: RETURN ELSE NC=5 :: RETURN
1260 IF NA=4 AND CTOT-TOT<=-20 THEN IF P< 90 THEN NC=3 :: RETURN ELSE NC=4 :: RE
TURN
1270 IF NA=4 AND CTOT-TOT>=20 THEN IF P< 80 THEN NC=3 :: RETURN ELSE NC=4 :: RET
URN
1280 IF NA=4 THEN IF P< 85 THEN NC=3 :: RETURN ELSE NC=4 :: RETURN
1290 IF NA=5 AND CTOT-TOT<=-20 THEN IF P< 95 THEN NC=4 :: RETURN ELSE NC=5 :: RE
TURN
1300 IF NA=5 AND CTOT-TOT>=20 THEN IF P< 85 THEN NC=4 :: RETURN ELSE NC=5 :: RET
URN
1310 IF NA=5 THEN IF P< 90 THEN NC=4 :: RETURN ELSE NC=5 :: RETURN
1898 REM
1900 REM ********************
1905 REM *   SUBROUTINES    *
1910 REM *     RICERCA      *
1915 REM *     SCHEMI       *
1920 REM ********************
1925 REM
1930 REM
1935 REM
1940 REM ********************
1945 REM  SUBROUTINE 2000
1950 REM  SCHEMI A 5 NUM.
1955 REM ********************
2000 FOR C1=1 TO C-8 !(C-1-3-4)
2010 LET W=C1
2020 LET Z=5
2030 IF X(C-1)=X(C1+4)AND X(C-2)=X(C1+3)AND X(C-3)=X(C1+2)AND X(C-4)=X(C1+1)AND
X(C-5)=X(C1)THEN GOSUB 6000
2040 NEXT C1
2050 IF C<>0 THEN GOSUB 7000 ELSE GOSUB 3000
2070 RETURN
2900 REM
2905 REM
2910 REM
2915 REM ********************
2920 REM  SUBROUTINE 3000
2925 REM  SCHEMI A 4 NUM.
2930 REM ********************
3000 FOR C2=1 TO C-7 !(C-1-3-3)
3010 LET W=C2
3020 LET Z=4
3030 IF X(C-1)=X(C2+3)AND X(C-2)=X(C2+2)AND X(C-3)=X(C2+1)AND X(C-4)=X(C2)THEN G
OSUB 6000
3040 NEXT C2
3050 IF C<>0 THEN GOSUB 7000 ELSE GOSUB 4000
3060 RETURN
3900 REM
3905 REM
3910 REM
3915 REM ********************
3920 REM  SUBROUTINE 4000
3925 REM  SCHEMI A 3 NUM.
3930 REM ********************
4000 FOR C3=1 TO C-5 !(C-1-2-2)
4010 LET W=C3
4020 LET Z=3
4030 IF X(C-1)=X(C3+2)AND X(C-2)=X(C3+1)AND X(C-3)=X(C3)THEN GOSUB 6000
4040 NEXT C3
4050 IF C<>0 THEN GOSUB 7000 ELSE GOSUB 5000
4060 RETURN
4900 REM
4905 REM
4910 REM
4915 REM ********************
4920 REM  SUBROUTINE 5000
4925 REM  SCHEMI A 2 NUM.
4930 REM ********************
5000 FOR C4=1 TO C-3 !(C-1-1-1)
5010 LET W=C4
5020 LET Z=2
5030 IF X(C-1)=X(C4+1)AND X(C-2)=X(C4)THEN GOSUB 6000
5040 NEXT C4
5050 IF C<>0 THEN GOSUB 7000 ELSE GOSUB 5500
5060 RETURN
5400 REM
5410 REM
5420 REM
5430 REM
5440 REM ********************
5450 REM  SUBROUTINE 5500
5460 REM  SCHEMI A 1 NUM.
5470 REM ********************
5500 FOR C5=1 TO C-2 !(C-1-1)
5510 LET W=C5
5520 LET Z=1
5530 IF X(C-1)=X(C5)THEN GOSUB 6000
5540 NEXT C5
5550 IF C<>0 THEN GOSUB 7000 ELSE LET NA=0
5560 RETURN
5900 REM
5905 REM
5910 REM
5915 REM
```

da nessun numero. C'è da aggiungere ancora che se veramente il numero previsto non è giusto, in questo caso è diverso da 2, l'1 non guadagna più +3 punti, perché non sussiste più il taglio. L'ultima considerazione da fare è relativa alle condizioni di gioco contingenti. Nel caso sia in vincita (+ 20) il computer allargherà le sue possibili scelte, anche a 3 numeri, per rendere meno prevedibili le sue mosse e amministrare "saggiamente" il suo vantaggio. Nel caso sia in perdita (– 20) opterà per una tattica il più fruttuosa possibile.

Come si vede le considerazioni sono molteplici e tutte quante insieme concorrono alla calibrazione dei pesi adoperati. Vi è infine l'ultima possibilità che la ricerca degli schemi non abbia avuto i frutti sperati (questo capita sempre nei primi turni di gioco). In questo caso si procederà alla generazione di numeri casuali che saranno condizionati dalle condizioni di gioco contingenti.

Il programma passo, passo...

1-89 *Righe di commento.* Qualcuno obietterà che abbia ecceduto nei "commenti", ma, come giustamente predica **Personal Software** (N. 3, pag. 23, regola 1), un buon programmatore deve sapersi autodocumentare.

90-199 *Qui si esplicano le consuete procedure di dimensionamento e inizializzazione.* Da notare L'OPTION BASE 1 alla riga 93 che serve per inizializzare ad 1 l'indice di partenza dei vettori, che nel TI Extended BASIC è di norma 0. C'è da osservare inoltre alla riga 185, l'istruzione ON WARNING NEXT, altra specifica del TI Extended BASIC, che permette di controllare la segnalazione di WARNING in caso di errore nel dato di input, cosa che scombussolerebbe tutto il quadro di gioco. Il NEXT fa sì che venga ignorato l'errore.

200-850 *Visualizzazione del quadro di gioco.* Dapprima si richiede al giocatore se vuole conoscere le regole del gioco e in quanti turni desidera giocare (da 50 a 200 turni). Poi, dopo aver disegnato il quadro di gioco, attende il numero dell'av-

Seguito listato 1.

```
5920 REM ******************
5925 REM  SUBROUTINE 6000
5930 REM CARICAMENTO(Y(C6))
5935 REM ******************
6000 LET C6=C6+1
6010 LET Y(C6)=X(W+Z)
6020 RETURN
6900 REM
6905 REM
6910 REM
6915 REM ******************
6920 REM  SUBROUTINE 7000
6925 REM  SCELTA DI -NA-
6930 REM ******************
7000 LET CAS3=INT(RND*C6)+1
7010 LET NA=Y(CAS3)
7020 RETURN
7030 REM
7040 REM
8900 REM ******************
8910 REM ******REGOLE******
8920 REM ******************
9000 CALL SCREEN(11)
9010 CALL CLEAR
9020 PRINT "    Le regole del Taglio:"
9030 PRINT
9040 PRINT "Questo e' un gioco strategi co, e consiste nel trvare la giusta stra
tegia psicologi ca per confondere e sopraffa re l'avversario(Computer) "
9050 PRINT "Si gioca in 50<>200 turni.A ogni turno dovete scegliere un numero(In
tero da 1 a 5) e inserirlo nel computer;il"
9060 PRINT "computer a sua volta ne pen sera' uno.A questo punto si  confrontera
nno i due numeri"
9070 PRINT "Se la loro differenza sara' diversa da 1, allora ognuno sommera' il
suo numero al proprio punteggio, se invece"
9080 PRINT "la differenza sara' esattamente 1, allora il giocatore  con il piu'
basso dei numeri li sommera' entrambi.mentre"
9090 PRINT "l'altro verra' -TAGLIATO- "
9099 CALL KEY(0,K,S)
9100 IF S<>0 THEN 9099 ELSE CALL CLEAR :: RETURN
9500 DATA 5,28,6,27,6,26,7,25,8,25,8,24,9,23,10,22,10,21,11,20,12,20,13,19,13,18
,14,17
9510 DATA 14,16,15,15,15,14,14,13,13,12,12,11,13,10,14,10,15,9,16,8,17,7,17,6,18
,5,19,4,-1,-1
9900 REM ******************
9910 REM  SUBROUTINE DELAY
9920 REM ******************
10000 SUB DELAY(SEC)
10010 FOR COUNT=1 TO 200*SEC
10020 NEXT COUNT
10030 SUBEND
20000 END
```

versario (N) e, solo ad inserimento avvenuto, provvederà a sua volta a "pensarne" uno (appare la scritta "COMPUTER THINKING") e visualizzarlo nella giusta posizione (righe 200-520). Si procede poi al conteggio e alla eventuale segnalazione di un taglio (righe 530-600). Alla fine dei turni stabiliti in precedenza, compariranno il punteggio finale e i soliti messaggi scherzosi di commiato. Inoltre si darà l'opportunità di giocare una nuova partita (righe 650-850).

Dalla riga 500 si accede alle subroutine per la ricerca degli schemi. La riga 510 provvede all'opportuno azzeramento del contatore della subroutine 6000. Le istruzioni ACCEPT-AT consentono di accettare dati da qualsiasi parte dello schermo e inoltre, cosa molto importante, controllano i dati in entrata, rifiutandoli se non sono conformi alle caratteristiche specificate nella VALIDATE. Le CALL DELAY si riferiscono al sottoprogramma DELAY (righe 10000-10030) e provocano delle pause lunghe tanti secondi, quanti specificati nell'argomento tra parentesi.

Dalla riga 900 cominciano le subroutine.

1000 *Ricerca strategica.* Prima di tutto viene incrementato il contatore dei turni (C) e viene inserito nel vettore X il numero scelto dal giocatore (N); poi sono generati due numeri casuali (CAS1 e CAS2) e, se sono stati giocati più di 9 turni, si salta la "strategia iniziale", riservata appunto ai primi 9 turni che sono casisticamente più facili da controllare. Dalla riga 1080 si effettua la ricerca strategica vera e propria. Per ottenere i pesi di cui sopra riferito ho provveduto alla generazione di numeri casuali. Per fare in modo, per esempio, che l'1 avesse il 65% di probabilità di essere scelto nei confronti del 5 (35%) è stata sufficiente la seguente condizione logica:

IF P < .65 THEN NC = 1 ELSE NC = 5
dove P = RND.

Quando invece si è trattato di "pesare" tre numeri allora si è dovuto sdoppiare l'istruzione per non creare una riga troppo complessa.

2000, 3000, 4000, 5000, 5500 *Qui si procede alla ricerca degli schemi numerici rispettivamente da 5, 4, 3, 2 e 1 numero.*

Ogni subroutine consta di un loop e di una condizione logica finale. All'interno del loop vi sono tre istruzioni; la terza è una istruzione logica che controlla numero per numero se lo schema è verificato; in tal caso richiama la subroutine 6000 e individuato con le prime due istruzioni il numero seguente lò schema lo si inserisce nel vettore Y. All'uscita del loop, quando tutti gli schemi sono stati provati, viene verificato se il vettore Y sia stato riempito. Nel caso sia vuoto si passa alla subroutine ricerca schemi successiva, altrimenti viene richiamata la subroutine 7000.

6000, 7000 *Accumulazione e scelta di NA.* La subroutine 6000 viene richiamata ogni qualvolta è stato verificato uno schema ed è quindi necessario conservare il valore di NA nel vettore Y. La subroutine 7000, una volta terminata la ricerca degli schemi, estrae casualmente dal vettore Y un valore di NA.

È importante ricordare che per far girare questo programma è necessario il modulo SSS TI Extended BASIC senza il quale sarebbe stato forse impossibile redigere un programma di questo genere, non possedendo il TI BASIC normale la fondamentale istruzione logica IF-THEN-ELSE estesa.

Conclusioni

È stato entusiasmante accorgersi come il computer, dopo un avvio non troppo brillante, iniziava a "fiutare" i primi schemi numerici e ad operare i primi tagli decisivi. È inevitabile che ciò accada perché prima o poi più o meno tutti siamo portati inconsciamente a schematizzare le nostre azioni secondo una logica che non sempre siamo in grado di rilevare a livello conscio. Il computer fa pressochè totale affidamento proprio su questa debolezza umana.

Non è comunque questo uno di quei casi in cui si può parlare del computer come di un giocatore imbattibile, ma vi posso assicurare che è un avversario tenace, soprattutto se vi mettete in testa di tendergli qualche tranello. Con lui potrete intrattenere delle interessanti partite.

In definitiva mi sembra di poter dire di avere ottenuto un programma capace di giocare, per lo meno alla pari, con il suo avversario. Può senz'altro essere migliorato e soprattutto modificato in molte parti con particolare riferimento alla strategia di gioco che, a mio avviso, può essere "interpretata" in diversi modi. Invito quindi i lettori a dare la loro interpretazione e a sfidare, in una battaglia computer contro computer, il mio programma. ■

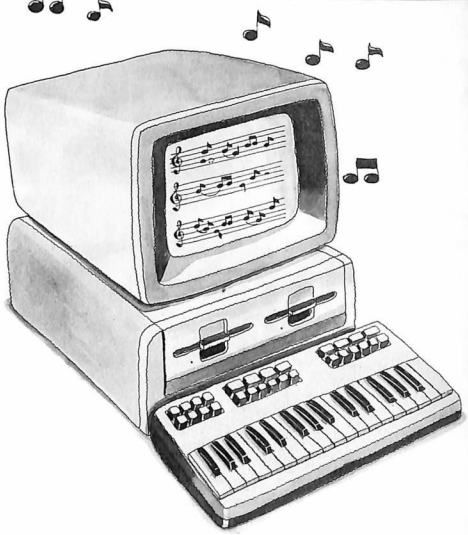
VARIABILI	DESCRIZIONE
X (Max. 200)	*Vettore principale (conserva i valori di N)*
Y (Max. 200)	*Vettore secondario (conserva i valori di NA)*
C	*Contatore vettoriale (X)*
NC	*Numero del computer*
NA	*Numero dell'avversario (probabile)*
CAS1, CAS2, CAS3	*Numeri casuali*
P (RD)	*Numero casuale per generare i pesi*
C1, C2, C3, C4, C5, C6	*Contatori delle subroutine*
TOT	*Totale del giocatore*
CTOT	*Totale del computer*
T	*Numero dei turni*
W, Z, A$	*Variabili di lavoro*
N	*Numero del giocatore*

Tabella 1. *Variabili del programma.*

PROBABILE NUMERO AVVERSARIO	NUMERO COMPUTER
	Normali
1	5 (60%) – 4 (40%)
2	1 (65%) – 5 (35%)
3	2 (75%) – 5 (25%)
4	3 (85%) – 4 (15%)
5	4 (90%) – 5 (10%)
	In perdita (–20)
1	5 (70%) – 4 (30%)
2	1 (70%) – 5 (30%)
3	2 (80%) – 5 (20%)
4	3 (90%) – 4 (10%)
5	4 (95%) – 5 (5%)
	In vincita (+20)
1	5 (40%) – 4 (35%) – 3 (25%)
2	1 (50%) – 5 (30%) – 4 (20%)
3	2 (55%) – 5 (30%) – 3 (15%)
4	3 (80%) – 4 (20%)
5	4 (85%) – 5 (15%)

Tabella 2. *Pesi per la strategia di gioco, espressi in percentuali. Il programma sceglie il proprio numero (NC) in funzione delle condizioni di gioco (NORMALI, IN PERDITA, IN VINCITA) e sulla base del probabile numero dell'avversario (NA).*

Musica con TI99/4A



Abbiamo ricevuto da Dario Paganini di Varese la conversione per TI99/4A del programma "Comporre musica con l'Atari" pubblicato sul n. 5. Il funzionamento è analogo a quello del programma originale.

Le linee da 10 a 80 stabiliscono (come da tabella) le corrispondenze fra i tasti e la frequenza delle note, usando le istruzioni READ/DATA.

I passi 1000/1110 sono relativi alla solita CALL KEY. Se non è stato premuto nessun tasto il puntatore S è 0 e viene riletta la tastiera. Se si è premuto un tasto corrispondente ad una nota, il programma pone il valore opportuno nella variabile TONE e (piccola modifica rispetto al programma Atari) riproduce quest'ultima al passo 1105 con la CALL SOUND. Se si è premuto un tasto richiedente un accordo il programma va alla routine che lo forma.

Ho aggiunto un confronto IF-THEN ai passi 2000/2100/2200/2300 per evitare che il programma si fermasse nel caso che il primo tasto premuto fosse un richiamo ad una routine per la formazione di un accordo. In questo caso infatti essendo la variabile TONE = 0 la CALL SOUND non potrebbe essere eseguita in quanto leggerebbe una frequenza 0 e il calcolatore darebbe il messaggio di errore "BAD VALUE IN...".

I valori di durata e volume della CALL SOUND possono essere facilmente variati secondo il gusto personale.

NOTA	TASTO	ASCII	FREQUENZA
LA	N	78	440
LA	J	74	466
SI	M	77	494
DO	Z	90	262
DO	S	83	277
RE	X	88	294
RE	D	68	311
MI	C	67	330
FA	V	86	349
FA	G	71	370
SOL	B	66	392
SOL	H	72	415
maggiore	1	49	—
settima	2	50	—
minore	3	51	—
min. 7ª	4	52	—

Tabella 1. *Corrispondenze fra tasti, codici ASCII, frequenze.*

```
10    !!!! REALTONE (12,2)
20    LASTBYTE=0
30    FOR I=1 TO 12
40    READ A,B
50    REALTONE (I,1)=A
60    REALTONE (I,2)=B
70    NEXT I
80    DATA 78,440,74,466,77,494,90,262,
           83,277,88,294,68,311,67,330,
           86,349,71,370,66,392,72,415
1000 CALL KEY(0,K,S)
1010 IF S=0 THEN 1000
1020 IF K=49 THEN 2000
1030 IF K=50 THEN 2100
1040 IF K=51 THEN 2200
1050 IF K=52 THEN 2300
1060 LASTBYTE=K
1070 FOR I=1 TO 12
1080 IF K<>REALTONE(I,1) THEN 1100
1090 TONE=REALTONE(I,2)
1100 NEXT I
1105 CALL SOUND(300,TONE,5)
1110 GOTO 1000
2000 IF TONE=0 THEN 1000
2010 CALL SOUND(300,TONE,5,INT(TONE/
     .79394+.5),5,INT(TONE/.66837+.5),5)
2020 GOTO 1000
2100 IF TONE=0 THEN 1000
2110 CALL SOUND(300,TONE,5,INT(TONE/
     .79394+.5),5,INT(TONE/.5625+.5),5)
2120 GOTO 1000
2200 IF TONE=0 THEN 1000
2210 CALL SOUND(300,TONE,5,INT(TONE/
     .84244+.5),5,INT(TONE/.66837+.5),5)
2220 GOTO 1000
2300 IF TONE=0 THEN 1000
2310 CALL SOUND(300,TONE,5,INT(TONE/
     .84244+.5),5,INT(TONE/.5625+.5),5)
2320 GOTO 1000
2400 END
```

Operatori logici per TI99/4A

Il signor Giuseppe Devoto di Ferrara ci manda questa nota su un metodo da lui escogitato.

Si tratta di un metodo per sopperire alla mancanza degli operatori logici che non sono presenti nel linguaggio base del calcolatore.

Il metodo che ho usato, non interviene direttamente sul linguaggio con procedure hardware, ma si basa su due proprietà:

1) il computer associa alle espressioni relazionali i valori numerici 0 e −1, rispettivamente, a seconda che siano "false" o "vere";
2) gli statement IF A<>O THEN ... e IF A THEN vengono considerati identici.

Vediamo come, sfruttando queste due proprietà, si possano realizzare dei surrogati degli operatori logici fondamentali, e cioè: And, Or e Not.

Iniziamo con l'operatore And, definito secondo la figura 1.

Il metodo che io utilizzo consiste nello scrivere, ad esempio, IF (A>B) * (B<C) THEN... che risulta equivalente a IF A>B AND B<C THEN...

Infatti: l'enunciato IF... THEN... esegue il salto condizionato alla riga indicata soltanto se l'espressione (A>B) * (C<D) fornisce un risultato diverso da zero, una volta che alle espressioni relazionali si sia sostituito il valore numerico corrispondente.

Se entrambe le relazioni sono "vere", il computer associa il valore numerico −1 a ciascuna di esse e il risultato complessivo è 1.

Se, invece, una delle due è falsa, il computer vi associa il valore 0 e il risultato complessivo è 0.

Nel primo caso il salto condizionato viene eseguito, nel secondo no.

Questo è esattamente il funzionamento richiesto per l'operatore logico And.

Si noti che non c'è alcun limite al numero di espressioni relazionali che l'operatore può legare; si può infatti scrivere: IF (A>B) * (B<C) * ... * (D=F) THEN...

Anche in questo caso il salto condizionato viene eseguito solo se sono vere tutte le espressioni indicate.

Per l'operatore Or, definito secondo la figura 2, si procede seguendo lo stesso principio.

Lo statement:

```
IF (A>B) + (C<D) THEN...
```

sostituisce:

```
IF A>B OR C<D THEN...
```

x	y	x and y
Vera	Vera	Vera
Vera	Falsa	Falsa
Falsa	Vera	Falsa
Falsa	Falsa	Falsa

Figura 1. *Operatore And.*

x	y	x or y
Vera	Vera	Vera
Vera	Falsa	Vera
Falsa	Vera	Vera
Falsa	Falsa	Falsa

Figura 2. *Operatore Or.*

x	y	x nand y
Vera	Vera	Falsa
Falsa	Vera	Vera
Vera	Falsa	Vera
Falsa	Falsa	Vera

Figura 3. *Operatore Nand.*

Infatti, esattamente come prima, se le espressioni relazionali sono entrambe false, ad esse viene associato il valore 0 ed è pure 0 il valore della loro somma. In queste condizioni il salto condizionato non viene eseguito.

Se, viceversa, almeno una delle espressioni relazionali è vera, il computer vi associa il valore −1 e la somma risulta diversa da zero. In questo caso il salto condizionato viene eseguito.

Anche in questo caso non esiste alcun limite al numero di espressioni relazionali che possono venire legate da un operatore.

Per quanto riguarda l'operatore Not, il discorso è lievemente più delicato. Cercherò di chiarire con un esempio i problemi che si presentano.

Supponiamo di voler realizzare l'operatore logico Nand, definito mediante la figura 3. Ricordiamo che un Nand viene anche definito come un And seguito da un Not.

Basterà allora scrivere IF ((A>B) * (B<C) −1) THEN...

Infatti: se A>B e B<C sono entrambe vere, si ha ((A>B) * (B<C) −1) = (−1); * (−1) −1 = 1−1 = 0 e il salto condizionato non viene eseguito.

In ogni altro caso si ha 0 − 1 = −1 e il salto viene eseguito. Si vede che il comportamento complessivo è quello richiesto dall'operatore Nand.

Il problema è questo: (A>B) * (B<C) vale 1, se entrambe le espressioni relazionali sono vere, perché è il prodotto di un numero pari di espressioni relazionali vere, e cioè di un numero pari di −1.

Si capisce come il risultato sarebbe −1 se il numero delle espressioni relazionali fosse dispari.

In questo caso, al prodotto di un numero dispari di espressioni relazionali, non si deve sottrarre, ma bensì aggiungere 1. Quindi, se si vuole negare un And, si deve aggiungere al prodotto il numero 1, se le espressioni relazionali sono in numero dispari, mentre si deve sottrarre 1 se le espressioni relazionali sono in numero pari.

Oltre a ciò non c'è alcun problema a realizzare il Not.

Volendo negare un Or il discorso si fa più delicato. L'espressione (A>B) + (B<C) può assumere i valori 0, −1, −2. Inoltre il valore non può venire previsto a priori: dipende dai valori di A, B e C.

Per rimediare si può usare la funzione segno: SGN ((A>B) + (B<C)) assume i soli valori 0 e −1 a seconda che l'espressione A>B OR B<C sia falsa o vera.

A questo punto, per effettuarne la negazione, si procede come prima: IF 1 + SGN ((A>B) + (B<C)) THEN...

Concludo facendo notare come le tecniche illustrate permettano di estendere le applicazioni del computer verso il campo delle variabili Booleane.

Infatti è possibile, con le tecniche illustrate e con un po' di attenzione, definire variabili numeriche che assumano i valori 0 e −1 ed utilizzarle in espressioni complesse dell'algebra di Boole.

Ad esempio:

```
A = (B>C) * (C<>D) * (C=F);
B = (1 + SGN ((A>C) + (D=F)));
H = A*B
```

e così via.

Naturalmente, per creare delle espressioni coerenti, è necessario prestare molta attenzione agli stessi problemi che avevamo visto per il Not: bisogna che le variabili booleane siano sempre ricondotte ai valori base 0 e −1, per esempio, attraverso un attento uso della funzione segno.

DEBUG

Guida e cacciatore

Il signor Aurelio Fantone scrive per segnalare alcuni errori riscontrati nel programma "Guida e cacciatore" per ZX81 pubblicato su **Personal Software** *8-9.*

Alla linea 85 del listato 1 si legge:

```
LET X = Q
```

ovviamente, poiché sia garantita l'impossibilità di "sparare" due numeri uguali consecutivamente, la linea andrà coì corretta:

```
LET Q = X
```

Inoltre, nello stesso programma, occorrerà inserire una istruzione che impedisca di porre 0 (zero) quale "numero da raggiungere" (22 IF Z = 0 THEN GOTO 15). In caso contrario infatti, già alla prima "mossa" con un input (diviso) si conclude banalmente il gioco. Similmente, nel listato n. 2 la linea 5 andrebbe così corretta:

```
LET Z = 1 + INT (RND × 100)
```

Sempre nel listato n. 2 è presente un altro errore: dando per esempio come primo input 1 e come secondo "sparo" 2, sul video appare:

```
2 + 2 = 2
```

Ciò è dovuto al fatto che con simili input nessuna delle condizioni (linee 60-65-70-75) viene a realizzarsi.

Si pone rimedio a quanto sopra, senza alterare la logica del programma, variando la linea 60 nel modo seguente:

```
60 IF A< = B AND A<=C AND A<= D THEN
   GOTO 200
```

Sistemi ridotti per il Totocalcio per C64

Molti lettori ci hanno segnalato malfunzionamenti del programma, al quale, in effetti, come segnalatoci dall'autore Carlo Sintini, manca la seguente istruzione:

```
155 T$(1) = "1": T$(2) = "X": T$(3) = "2"
```

PERSONAL SOFTWARE

I file su cassetta nel TI99/4A

Qualche utile suggerimento per organizzare i vostri archivi

di *Filippo Cerulo*

La gestione dei file su cassetta nel TI99/4A, rappresenta, forse, una delle maggiori difficoltà che incontrano molti utenti, soprattutto principianti.

I motivi di ciò vanno ricercati soprattutto nella impostazione particolare del manuale che accompagna la macchina: la sezione che riguarda i file è chiarissima dal punto di vista tecnico, (basta leggere con attenzione), ma è assolutamente priva di esempi di utilizzo, cosicchè una persona che ignori il significato e l'uso di file e record dopo mezza pagina salta magari al capitolo sulla grafica che è senz'altro più accessibile. Inoltre, dai programmi pubblicati su alcune riviste, potrebbe dedursi che anche utilizzatori più smaliziati hanno qualche dubbio su questo importante argomento. Allora forse sarà utile spendere qualche parola in più.

File e record

La traduzione italiana più frequente della parola file è archivio.

Infatti un file non è altro che un archivio, cioè un insieme di informazioni logicamente correlate, che in genere vengono conservate per successive consultazioni o aggiornamenti.

Ci sono infiniti esempi di archivio, anche molto vicini alla nostra esperienza comune. Sono archivi: l'agenda degli indirizzi, il libro delle ricette, l'estratto conto della banca ecc. Per fissare le idee, consideriamo un esempio banale: una rubrica telefonica.

Essa non è altro che un elenco di nomi e numeri telefonici come per esempio:

ROSSI MARCO	06/871190
VERDI ELIO	0823/771432
BIANCHI ALEX	082/919293

Se identifichiamo, come in questo caso, il file con una tabella costituita da tante righe contenenti informazioni dello stesso tipo, definiamo RECORD LOGICO del file, ogni riga della tabella.

Nel nostro caso un record logico è per es.: VERDI ELIO 0823/771432.

Ogni record è formato a sua volta da CAMPI. I record del nostro archivio sono formati da tre campi:

CAMPO N.	NOME DEL CAMPO	CONTENUTO
1	"cognome"	Verdi
2	"nome"	Elio
3	"numero"	0823/771432

Le operazioni che normalmente si compiono su un archivio (la nostra rubrica telefonica) sono:

1) *Creazione dell'archivio:* quando copiate sull'agenda nuova i numeri che volete conservare.
2) *Aggiunta di un record:* quando scrivete il numero di quella tedesca bionda conosciuta al mare.
3) *Cancellazione di un record:* quando decidete che Carlo è veramente antipatico ed il suo numero è meglio non averlo.
4) *Modifica di un record:* quando il vostro medico di fiducia cambia casa e quindi numero di telefono.

In generale queste sono anche le operazioni che si compiono su qualunque tipo di archivio. L'organizzazione di un archivio dipende in parte dalle informazioni che esso deve contenere, in parte dalle operazioni che volete compiere sui record che lo costituiscono. Ad esempio, i record logici della nostra rubrica telefonica potrebbero anche avere solo due campi:

CAMPO N.	NOME DEL CAMPO	CONTENUTO
1	"nome"	Verdi Elio
2	"telefono"	0823/771432

Ciò vuol dire che una volta scelte le informazioni che dobbiamo archiviare, potremo organizzarle in record di tipo diverso a seconda delle nostre esigenze.

Un'altra operazione fondamentale che di solito si compie sui file è l'ordinamento. Esso in generale può essere fatto secondo qualunque campo del record. Ad esempio potremo ordinare i nomi della nostra rubrica alfabeticamente oppure per numeri telefonici crescenti.

Esistono svariati metodi per ordinare un file, ma su di essi non mi soffermerò. Mi preme solo ribadire che il file è una struttura usata soprattutto per conservare le informazioni a computer spento e come tale ha un'importanza fondamentale. Vi consiglio perciò di approfondire l'argomento su testi specifici o su un buon manuale di programmazione. In ogni caso altri esempi saranno esaminati in seguito.

Il TI99/4A e i file

A differenza di molti altri computer della sua classe, il nostro gestisce i file in modo notevolmente sofisti-

I file su cassetta nel TI99/4A

cato, sicuramente all'altezza di macchine più potenti e costose. Soprattutto con i DISK DRIVE offre possibilità molto ampie, tali da garantire una corretta e facile gestione di grandi masse di dati.

Nel seguito però non si parlerà dei DISK DRIVE (alzi la mano chi ne possiede uno) ma del più semplice (ma non meno utile) registratore a cassette. Tornando al nostro esempio, la rubrica telefonica, supponiamo di voler conservare i numeri di dieci amici che abbiamo memorizzato in due vettori di dieci elementi: NOM$(I) e TEL$(I). Il puntatore I indica la corrispondenza tra nomi e numeri. Così, ad esempio, per I = 3 si avrà:

```
NOM$(3) =   BIANCHI ALEX
TEL$(3) =     082/919293
```

Il nostro record logico è dunque formato da due campi che abbiamo chiamato NOM$ (I) e TEL$ (I).

Come su molti altri sistemi, prima di utilizzare un file dobbiamo in qualche modo informare la macchina delle nostre intenzioni; è questo il senso della istruzione di "apertura file" che sul TEXAS ha la seguente sintassi:

OPEN # numero file: nome dispositivo, organizzazione file, tipo file, modo di apertura, tipo record.

Esaminiamo un po' più da vicino i parametri da definire:

Numero di file: può essere un intero tra 1 e 255. Serve ad identificare nelle istruzioni del programma il file a cui facciamo riferimento.

Nome dispositivo: identifica il registratore che vogliamo usare: può essere "CS1" o "CS2".

Organizzazione file: con le cassette può essere solo SEQUENTIAL cioè sequenziale. Questa dichiarazione può essere omessa una volta specificato l'uso di un registratore.

Modo di apertura: INPUT se vogliamo leggere dati dal nastro; OUTPUT se invece vogliamo scrivere dati sul nastro. Ricordate che il "CS2" cioè il secondo registratore, non può leggere dati ma solo scriverli. Così se il nome dispositivo è "CS2", il modo di apertura può essere solo OUTPUT.

Tipo file: INTERNAL o DISPLAY; questo è un punto importante su cui mi soffermerò in seguito. Se non specificate il "tipo file", il calcolatore lo assume DISPLAY.

Tipo record: per le cassette può essere solo FIXED cioè possiamo usare solo record FISICI di lunghezza fissa. Possiamo però scegliere tra record lunghi 64, 128 o 192 caratteri. Quando non si specifica la lunghezza del record, essa viene assunta uguale a 64 caratteri. Non sottovalutate l'importanza di questa scelta: in molti casi può risultare addirittura determinante per l'efficienza dei programmi.

Una volta utilizzato il file, per informare il calcolatore che deve chiudere il "canale di comunicazione" con il dispositivo esterno, è disponibile una semplice istruzione:

CLOSE # numero di file

dove "numero di file" è il numero, assegnato nella OPEN, del file che vogliamo chiudere. Naturalmente ci sono ancora le istruzioni più importanti, quelle che ci permettono di leggere e scrivere dati sul nastro.

Esse sono rispettivamente:

INPUT # numero di file: lista di variabili.

PRINT # numero di file: lista di variabili.

A parte il numero di file, esse hanno una sintassi molto simile alla PRINT ed alla INPUT normalmente usate nei vostri programmi. Il motivo è che il TI99/4A gestisce come file anche la tastiera ed il video ed assegna loro il numero di file uguale a 0. Infatti le istruzioni:

```
100 PRINT "CIAO CIAO"
```

e

```
100 PRINT # 0 : "CIAO CIAO"
```

oppure

```
110 INPUT A$
```

e

```
110 INPUT # 0 : A$
```

sono perfettamente equivalenti. Tornando al nostro esempio, supposto acquisito precedentemente il contenuto dei vettori NOM$ (I) e TEL$ (I) (cioè il nome ed il numero di telefono dei nostri amici) per registrarlo su nastro potremmo scrivere le seguenti istruzioni:

```
.....
300 OPEN # 1 : "CS1", INTERNAL, OUTPUT, FIXED
310 FOR I = 1 TO 10
320 PRINT # 1 : NOM$ (I), TEL$ (I)
330 NEXT I
340 CLOSE # 1
.....
```

E, qualora volessimo rileggerlo:

```
.....
500 OPEN # 2 : "CS1", INTERNAL, INPUT, FIXED
510 FOR I = 1 TO 10
520 INPUT # 2 : NOM$ (I), TEL$ (I)
530 NEXT I
540 CLOSE # 2
.....
```

Notate come sia necessaria una perfetta corrispondenza dell'ordine di scrittura con quello di lettura, ciò per evitare che variabili diverse si scambino i contenuti. Sia per i file INTERNAL che per quelli DISPLAY, le variabili da stampare su un record devono essere separate da una virgola, un punto e virgola o due punti. Quando la lista di variabili termina con uno di questi separatori, la successiva istruzione riguardante il file fa riferimento al record precedente.

I file su cassetta nel TI99/4A

Nel nostro esempio, ad ogni ciclo del FOR stampavamo o leggevamo le variabili NOM$(I) e TEL$(I) in un singolo record. Ma le istruzioni:
320 PRINT # 1 : NOM$(I), TEL$(I)
e
320 PRINT # 1 : NOM$ (I),
325 PRINT # 1 : TEL$ (I)
oppure
520 INPUT # 1 : NOM$ (I),
525 INPUT # 1 : TEL$ (I)
e
520 INPUT # 1 : NOM$ (I), TEL$ (I)
sono perfettamente equivalenti: siccome la 320 PRINT # 1 : NOM $ (I), e la 520 INPUT # 1 : NOM$(I), terminano con un separatore, in questo caso la virgola, allora la successiva istruzione fa riferimento allo stesso record.

Abbiamo però che, pur esatte formalmente, le parti di programma che avete appena letto sono quanto di peggio si possa scrivere almeno nel nostro caso. Bisogna infatti tenere presenti alcune piccole particolarità di cui parleremo adesso.

Internal o display?

Le differenze fondamentali tra i due tipi di file sono:

1) DIVERSA RAPPRESENTAZIONE DEI DATI. Quelli di tipo INTERNAL sono scritti con la codifica interna del computer; quelli di tipo DISPLAY sono invece scritti con i normali caratteri ASCII standard.

Questo perchè il TI99/4A gestisce con la tecnica del file quasi tutti i dispositivi esterni collegabili.

Tra i "nomi dispositivo" specificabili c'è infatti anche "RS 232" cioè l'interfaccia seriale che di solito unisce una stampante al computer. Dunque per stampe cartacee è indispensabile il formato display.

2) DIVERSO FORMATO DI STAMPA.

Per il tipo *INTERNAL*: I dati numerici sono *sempre* lunghi 9 caratteri. Quelli alfanumerici sono lunghi quanto la stringa da stampare più uno. Ecco alcuni esempi:

DATO	TIPO	LUNGHEZZA
12	numerico	9
12	alfanumerico	3
PIPPO	alfanumerico	6
9876543294	numerico	9

Il primo carattere di ogni dato alfanumerico indica la lunghezza del dato stesso.

Attenzione: se il numero totale dei caratteri delle variabili da stampare nel record supera la capacità del record stesso, si ha un messaggio di errore ed il programma si ferma. Dobbiamo quindi valutare in anticipo il numero di caratteri massimo che può raggiungere la nostra lista di variabili da stampare in ogni record. Così ad esempio in un record *fisico* lungo 64 caratteri possiamo registrare al massimo 7 dati numerici.

Per il tipo DISPLAY: la lunghezza del dato è uguale al numero di caratteri che lo formano. Esempi:

DATO	TIPO	LUNGHEZZA
12	numerico	3
12	alfanumerico	2
PIPPO	alfanumerico	5
9876543294	numerico	10

Essi devono però al momento della stampa, essere separati da una virgola che viene inserita nella lista di stampa.

Volendo registrare i record del nostroesempio in formato DISPLAY, avremmo dovuto scrivere:

```
300 OPEN # 2 : "CS1", DISPLAY,
    OUTPUT, FIXED
310 FOR I = 1 TO 10
320 PRINT # 2 : NOM$(I);",";TE-
    L$(I);","
330 NEXT I
340 CLOSE # 2
```

e, per rileggerli dovremmo semplicemente cambiare le istruzioni 300 e 320 in:

```
300 OPEN # 2 : "CS1", DISPLAY,
    INPUT, FIXED
320 INPUT # 2 : NOM$(I);",";TE-
    L$(I);","
```

Se la lunghezza delle variabili da scrivere nel record supera la capacità fissata del record stesso, non si ha un FILE ERROR, ma l'ultimo dato diventa il primo del record successivo. Per evitare errori di lettura, tuttavia, è bene che anche in questo caso si valuti attentamente il numero di caratteri delle variabili da stampare.

Nelle normali operazioni di lettura/scrittura da nastro, consiglio senz'altro il formato INTERNAL. Ma resta da stabilire ... *quanti caratteri dopo il fixed*?

Abbiamo già definito un record logico: esso è l'unità logica di informazione di un archivio. Abbiamo però parlato anche di RECORD FISICO: nel nostro caso è la porzione fisica di nastro su cui vengono registrate le informazioni. Ogni volta che scrive un RECORD FISICO, il nostro computer si ferma per qualche attimo (o meglio ferma il registratore). Noi abbiamo la possibilità, attraverso l'istruzione OPEN, di stabilire quanti caratteri esso debba stampare per ogni record fisico (64, 128, 192). Tornando al nostro esempio, supposto che i nomi dei nostri amici siano lunghi al massimo 20 caratteri e riservando 10 caratteri per il numero di telefono, il nostro RECORD LOGICO è lungo al massimo 32 caratteri (infatti due caratteri indicano la lunghezza delle stringhe NOM$(I) e TEL$(I)).

Per come abbiamo scritto il programma, però, il record FISICO è lungo 64 caratteri. Rimangono perciò inutilizzati ben 32 caratteri.

La prima soluzione che ci viene in mente è registrare 2 record logici per ogni record fisico. Ma se consideriamo che la differenza di tempo di registrazione per ogni record tra un FIXED 64 e un FIXED 192 è di soli due secondi risulta lampante che la soluzione ottimale è registrare 6 record logici per ogni record fisico lungo 192 caratteri. In solo 8 secondi dunque stamperemo 6 dei nostri record logici contro i 6 secondi ne-

cessari a registrarne uno solo nel nostro primo programma. Ecco come potremmo fare:

```
200 OPEN # 2 : "CS1", INTERNAL, OUTPUT, FIXED 192
210 FOR I = 1 TO 12 STEP 6
220 FOR J = 1 TO I + 4
230 PRINT # 2 : NOM$ (J), TEL$ (J),
240 NEXT J
250 PRINT # 2 : NOM$ (J), TEL$ (J)
260 NEXT I
270 CLOSE # 2
```

e per rileggere:

```
300 OPEN # 2 : "CS1", INTERNAL, INPUT, FIXED 192
310 FOR I = 1 TO 12 STEP 6
320 FOR J = 1 TO I + 4
330 INPUT # 2 : NOM$ (J), TEL$ (J),
340 NEXT J
350 INPUT # 2 : NOM$ (J), TEL$ (J)
360 NEXT I
370 CLOSE # 2
```

Il ciclo di FOR sulla variabile I serve a dividere i due blocchi di registrazione, mentre il FOR sulla J (linee 220, 230, 240) serve a scrivere sullo stesso record FISICO cinque dei nostri record logici ognuno formato da un nome e da un numero di telefono.

Ciò è possibile in quanto l'istruzione 230 termina con una virgola.

L'istruzione 250 stampa il sesto record logico e chiude la stampa nel record fisico precedente perchè non termina con *alcun separatore*. Lo stesso discorso vale per la routine di input.

Il "segreto" dunque per una corretta gestione dei file è nello scegliere sempre una FIXED 192 ed "impacchettare" il maggior numero di informazioni possibili in 192 caratteri con un accorto uso dei separatori.

Conclusioni

Anche se ottimizzare programmi che gestiscono file può non sembrare facile, almeno a chi affronta per la prima volta simili problematiche, alla luce di quanto finora detto i punti da considerare non sono molti. Con un po' di pratica e di accortezza tutto sembrerà più facile e gratificante soprattutto per chi non si accontenta più solo di giochini e cerca di affrontare argomenti più "seri" ma non per questo meno divertenti. ■

È vero: piccolo è bello!

Alla scoperta dello ZX SPECTRUM

a cura di **Rita Bonelli**

ZX Spectrum è l'ultimo nato della famiglia Sinclair. È un calcolatore a colori di piccole dimensioni, ma di grandissime possibilità. Imparare a usarlo bene può essere fonte di molte piacevoli scoperte. Questo libro vi aiuta a raggiungere lo scopo. In 35 brevi e facilissimi capitoli non solo imparerete tutto sulla programmazione in BASIC, ma arriverete anche a usare efficientemente il registratore e a sfruttare al meglio le stampe. Soprattutto capirete la differenza tra il vostro Spectrum e gli altri computer.

320 pagine. Lire 22.000 Codice 337 B

GRUPPO EDITORIALE JACKSON

SCONTO 20% AGLI ABBONATI FINO AL 28-2-'84

Per ordinare il volume utilizzare l'apposito tagliando inserito in fondo alla rivista

TEXAS TI99/4A

Rette ad alta risoluzione

Sergio Borsani

Consideriamo brevemente l'output su video del TI99/4A.

Il computer utilizza 24 righe e 32 colonne; in ognuna di queste posizioni può essere posto un carattere. Ad esempio l'istruzione CALL HCHAR (12, 16, 42), può essere usata come comando immediato che nei programmi BASIC fa apparire un asterisco (codice ASCII 42) nella posizione individuata dalla riga 12 e dalla colonna 16. Ricordate che le righe sono numerate dall'alto verso il basso e le colonne da sinistra a destra. La superficie occupata da un carattere è composta da una matrice quadrata formata da 8 x 8 = 64 punti video (pixel). È possibile definire nuovi caratteri con l'istruzione CALL CHAR, specificando una stringa di 16 caratteri in codice esadecimale (vedi manuale), ognuno dei quali è associato ad un gruppo di 4 pixel. Ad esempio, CALL CHAR (128, "FF83858991A1C1FF") definisce un nuovo carattere che ha la forma di un quadratino attraversato da una diagonale; per farlo apparire bisogna usare CALL HCHAR; ricordate inoltre che i codici esadecimali posti tra virgolette in CALL CHAR vanno scritti con le lettere maiuscole.

Come molti sanno, l'equazione esplicita della retta è: $y = m\ x\ n$, dove m rappresenta il coefficiente angolare ed n l'ordinata all'origine. Semplificando, m determina l'incinazione della retta, n la fa spostare verso l'alto o verso il basso. Se n è uguale a 0 la retta passa per l'origine degli assi cartesiani.

Il modo più semplice per disegnare una retta con il computer è quello di stampare una serie di asterischi, uno per ogni colonna, usando l'istruzione CALL HCHAR. Molte versioni del BASIC sono sprovviste di tale comando, nel qual caso si ricorre al PRINT TAB.

Comunque, una retta rappresentata da una serie di asterischi sembra tale solo per certe inclinazioni: per $m = 0$ (retta orizzontale) ed $m = 1$ (retta a 45°); in tutti gli altri casi assume un poco gradito aspetto a gradini.

Questo può essere accettabile su monitor ad 80 colonne ma non lo è assolutamente su uno a 32.

Se non avete mai provato, fate girare il programma del listato 1.

Per ottenere una retta ad alta risoluzione bisogna essere in grado di gestire i singoli punti video. In alcuni computer è sufficiente il comando LINE (X_1, Y_1) – (X_2, Y_2) per ottenere un segmento di retta che congiunge due punti video.

Con il TI99/4A la cosa è possibile solo creando una routine di assembler e sappiamo che per disporre di tale linguaggio è necessario avere il sistema a dischi.

Apparentemente sembra impossibile ottenere lo stesso risultato in BASIC, il programma presentato in questo numero dimostra tuttavia il contrario.

Il prezzo pagato, però, è molto alto e si traduce in una lenta esecuzione, al punto che il programma non può essere usato per normali applicazioni ma va piuttosto considerato come un "pezzo da museo".

All'inizio dell'esecuzione viene chiesto di introdurre i vlori di m e di n. Il coefficiente angolare può assumere valori positivi o negativi, è opportuno tuttavia che non oltrepassi un certo limite; già per $m = 3$ i punti che formano la retta non sono più adiacenti tra di loro; perfezionando il programma sarebbe tuttavia possibile eliminare questo inconveniente. Anche n può assumere valori positivi o negativi, ma se questi sono troppo alti la retta cade fuori campo e non viene visualizzata. Alla riga 330 viene definita l'equazione della retta: y è l'ordinata, in numero di pixel, x è l'ascissa; n è stato moltiplicato per 10 in modo da ottenere una immagine "ingrandita" del grafico. Nelle linee 570-690 viene individuato il carattere più vicino all'asse verticale che viene attraversato dalla retta; NX ed NY sono il numero di caratteri interposti tra questo e gli assi.

Le linee 700-1070 contengono la routine che determina quali pixel, all'interno del carattere, devono esser messi in posizione ON; il compito è svolto da un ciclo interno FOR ... NEXT che esplora i pixel a gruppi di 4 e genera un codice esadecimale, e da un ciclo esterno FOR ... NEXT di 16 passi che compone la stringa atta a definire l'intero carattere.

La variabile MAX memorizza la distanza in pixel dell'ultimo punto dell'asse verticale man mano che procede la costruzione della retta, mentre T numera i caratteri che di volta in volta vengono visualizzati.

In sostanza il programma, non potendo gestire i punti video indipendentemente uno dall'altro, si incarica di esplorarli, raggrupparli, generare da questi una stringa in codice esadecimale, definire un carattere stamparlo quindi nella posizione opportuna.

Ogni ciclo completo viene effettuato in circa 11", tuttavia una retta può essere formata da più di 20 caratteri, il che comporta un tempo totale di esecuzione intorno ai 3 minuti, in verità troppo alto se si dovesse usare la routine in un programma applicativo.

Tengo a precisare che tale lentezza non è imputabile al TI99/4A, che in alcuni *benchmarks* da me effet-

Rette ad alta risoluzione

tuati si è generalmente dimostrato all'altezza di molti veri personal per quanto riguarda la velocità a livello di CPU, ma esclusivamente all'uso del BASIC come linguaggio di programmazione in questa particolare applicazione.

Sapreste, cambiando solo poche righe, ottenere grafici di altre funzioni? ■

Listato 1. *Esempio di retta rappresentata mediante una serie di asterischi.*

```
100 CALL CLEAR
110 PRINT TAB(4);"EQUAZIONE ";
120 PRINT "DELLA RETTA"
130 PRINT TAB(10);"Y=M*X+N"
140 INPUT "M = ":M
150 INPUT "N = ":N
160 DEF Y=M*X+N
170 CALL CLEAR
180 FOR X=1 TO 32
190 IF Y<1 THEN 220
200 IF Y>24 THEN 220
210 CALL HCHAR(25-Y,X,42)
220 NEXT X
230 END
```

Listato 2. *Programma retta ad alta risoluzione.*

```
100 REM ***************
110 REM *     RETTE     *
120 REM *    AD ALTA    *
130 REM * RISOLUZIONE *
140 REM ***************
150 CALL CLEAR
160 DIM A$(50)
170 T=1
180 MAX=0
190 PRINT "              RETTE"
200 CALL CHAR(96,"0101010101010101")
210 CALL CHAR(97,"FF")
220 CALL CHAR(98,"0101010101010701")
230 CALL CHAR(99,"0101010107010101")
240 CALL CHAR(100,"FF404")
250 CALL CHAR(101,"FF101")
260 CALL CHAR(104,"0")
270 FOR C=10 TO 16
280 CALL COLOR(C,2,16)
290 NEXT C
300 PRINT "            Y=M*X+N"
310 INPUT "COEFF. ANGOLARE? (M) ":M
320 INPUT "ORD. ALL'ORIGINE? (N) ":N
330 DEF Y=INT(M*X+N*10)
340 CALL CLEAR
350 FOR RIGA=3 TO 16
360 CALL HCHAR(RIGA,10,104,14)
370 NEXT RIGA
380 CALL HCHAR(17,10,97,14)
390 CALL VCHAR(3,9,96,14)
400 CALL HCHAR(18,8,48)
410 CALL HCHAR(17,16,100)
420 CALL HCHAR(18,16,53)
430 CALL HCHAR(17,22,101)
440 CALL HCHAR(18,22,49)
450 CALL HCHAR(18,23,48)
460 CALL HCHAR(10,9,98)
470 CALL HCHAR(10,8,53)
```

Seguito listato 2.

```
480 CALL HCHAR(4,9,99)
490 CALL HCHAR(4,7,49)
500 CALL HCHAR(4,8,48)
510 EQ$="Y=("&STR$(M)&")*X+("&STR$(N)&")"
520 IF LEN(EQ$)>20 THEN 570
530 FOR I=1 TO LEN(EQ$)
540 Z=ASC(SEG$(EQ$,I,1))
550 CALL HCHAR(20,9+I,Z)
560 NEXT I
570 X=1
580 IF X>112 THEN 1080
590 IF Y>112 THEN 620
600 IF Y<1 THEN 620
610 GOTO 640
620 X=X+1
630 GOTO 580
640 NY=INT(Y/8)
650 IF NY<>Y/8 THEN 670
660 NY=NY-1
670 NX=INT(X/8)
680 IF NX<>X/8 THEN 710
690 NX=NX-1
700 REM  *DEFINIZIONE E STAMPA CARATTERE*
710 J=8
720 NXX=8*NX
730 NYY=8*NY
740 FOR F=1 TO 16
750 S=0
760 Y1=NYY+J
770 IF INT(F/2)=F/2 THEN 800
780 W=0
790 GOTO 810
800 W=4
810 FOR K=1 TO 4
820 X=NXX+K+W
830 IF Y1=INT(Y)THEN 860
840 P=0
850 GOTO 890
860 P=2^(4-K)
870 IF X<=MAX THEN 890
880 MAX=X
890 S=S+P
900 NEXT K
910 IF W=0 THEN 930
920 J=J-1
930 IF S>9 THEN 960
940 COD$=STR$(S)
950 GOTO 970
960 COD$=CHR$(55+S)
970 IF F<>1 THEN 1000
980 CAR$=COD$
990 GOTO 1010
1000 CAR$=CAR$&COD$
1010 NEXT F
1020 A$(T)=CAR$
1030 CALL CHAR(104+T,A$(T))
1040 CALL HCHAR(16-NY,10+NX,104+T)
1050 T=T+1
1060 X=MAX+1
1070 GOTO 580
1080 V$="VUOI CONTINUARE? (S/N) "
1090 FOR I=1 TO LEN(V$)
1100 L=ASC(SEG$(V$,I,1))
1110 CALL HCHAR(24,3+I,L)
1120 NEXT I
1130 CALL KEY(0,KEY,STATUS)
1140 IF STATUS=0 THEN 1130
1150 CALL CLEAR
1160 IF KEY=83 THEN 150
1170 IF KEY=115 THEN 150
1180 END
```

TI99/4A HOME COMPUTER

Videografici

Sergio Borsani

In questo stesso numero, mostro come sia possibile, con una routine in BASIC, ottenere il grafico di una retta ad alta risoluzione, anche se i tempi di esecuzione sono molto lenti. Il maggiore inconveniente è che i pixel video, pur essendo indirizzabili individualmente, sono condizionati dagli altri punti-video che formano un carattere. Pertanto la prassi consiste nel definire un carattere indirizzando al suo interno i punti mediante una stringa in codice esadecimale (CALL CHAR(N,A$)) e poi indirizzare sul video tutto il carattere mediante una CALL HCHAR(X,Y,N), dove X è il numero di riga, Y il numero di colonna e N il numero del carattere precedentemente definito.

Per la grafica ad alta risoluzione è consigliabile munirsi di carta quadrettata, con quadrati suddivisi in 8 x 8 quadratini (purtroppo la carta millimetrata non è del tutto adatta). Come esempio immaginiamo di voler definire un carattere che riproduce il cavallo del gioco degli scacchi (vedi figura 1).

Il disegno va diviso in due parti con una linea verticale ed in tutto si ottengono 16 blocchi di quattro pixel ciascuno. I blocchi vanno letti, normalmente, da sinistra a destra e dall'alto in basso. Essi si traducono poi in codice esadecimale, il manuale (pag. 84) è abbastanza chiaro a tale proposito. Il disegno del nostro esempio viene definito con l'istruzione CALL CHAR(128,"083C3E0E1E1E0C3E").

La stringa che compare nell'istruzione è composta da 16 caratteri esadecimali, tanti sono i blocchi, tuttavia eventuali zeri finali possono essere tralasciati; ad esempio, volendo in posizione ON tutti i pixel di un carattere sarà sufficiente l'istruzione CALL CHAR(129,"F") e poi dare lo stesso colore al carattere ed allo sfondo con CALL COLOR(13,9,9). Il numero 9 si riferisce al rosso.

Le cose si complicano quando si vogliono creare immagini ad alta definizione composte da molti caratteri. In un mio programma ho riprodotto la cartina d'Italia, comprese le isole, in dimensioni tali da occupare quasi tutto lo schermo.

Per far questo ho dovuto definire circa 80 caratteri e posizionarli opportunamente sullo schermo. Quando si trattano caratteri ridefiniti in numero così elevato bisogna essere consapevoli che non possono essere utilizzati per altri scopi. Paradossalmente, se si ridefinisce il carattere con codice ASCII 32 (space), quando poi nel programma si dà l'istruzione CALL CLEAR, tutto lo schermo si riempie con il nuovo carattere invece di pulirsi.

Una volta definiti, i caratteri vanno indirizzati sul video. Se sono in gran numero non sarà necessario ripetere decine di volte l'istruzione CALL HCHAR, ma sarà da preferirsi l'uso di un ciclo FOR NEXT come nel seguente esempio:

```
...
500 DATA 12, 15, 128, 12, 16, 129 ...
510 RESTORE 500
520 FOR J = 1 TO ...
530 READ X, Y, N
540 CALL HCHAR (X,Y, N)
550 NEXT J
...
```

I dati devono essere nell'ordine: numero di riga, di colonna, di carattere.

In ogni caso la creazione di immagini composte da molti caratteri comporta sempre una notevole perdita di tempo. Sarebbe certamente vantaggioso disegnare direttamente su video, tramite la tastiera, lasciando al computer il compito di determinare la stringa di caratteri esadecimali.

Questo è quanto ho realizzato nel programma che presento nel listato 1.

Selezionato da menu il tasto 1, compaiono sul video un quadrato bianco ed un riquadro. Il primo rappresenta un carattere ingrandito, il secondo lo spazio in cui andrà collocato il disegno ad alta risoluzione che si vuole realizzare. Il quadrato bianco viene spazzato da un cursore. Premendo i tasti 1 e 0, il cursore lascia dietro di sè rispettivamente un punto nero (pixel ON) o un punto bianco (pixel OFF). Se si commette un errore si può annullare l'ultima parte del lavoro con SHIFT(−). Completata l'operazione appare la scritta PROCEDO? Ad una risposta affermativa compare il numero del carattere e la stringa in codice esadecimale che lo definisce, inoltre appare il carattere in dimensioni reali nel riquadro. (Figura 2).

Azionando i tasti muniti di freccia (E = alto, S = sinistra, D = destra, X = basso) si può quindi collocare quest'ultimo nella posizione più opportuna. Premendo (1) si procede alla definizione di un secondo carattere.

Continuando in questo modo si aggiungono nuovi caratteri e si accostano a quelli precedentemente definiti fino a formare una figura complessa ad alta risoluzione. Al termine bisogna premere SHIFT(+).

Se si limitasse a questo il programma avrebbe solo un valore didattico, invece esso permette di memorizzare su nastro (o, modificando le OPEN, su dischetto) tutta l'immagine creata per poterla poi richiamare e riutilizzare in un altro programma; anzi, in questo secondo programma, potrebbero trovar posto più

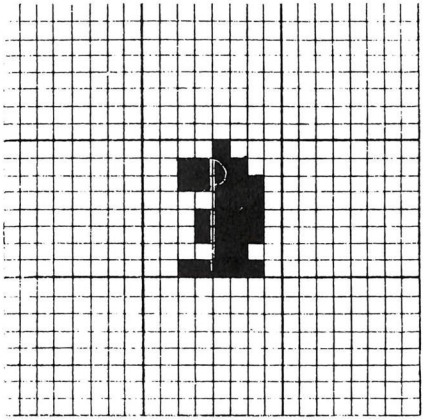


Figura 1. *Rappresentazione del cavallo nel gioco degli scacchi. Viene definito con l'istruzione CALL CHAR(128, "083C3E0E1E1E0C3E").*

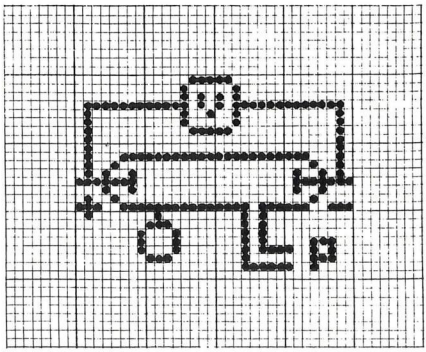


Figura 3. *Disegno in alta risoluzione formato da 12 caratteri. Con Videograf si creano immagini come questa, si memorizzano e si riutilizzano in altri programmi.*

disegni, richiamati successivamente con le stesse modalità. (Figura 3).

Nella prima fase la "finestra" che contiene il disegno viene esplorata con una istruzione CALL GCHAR che restituisce i numeri dei caratteri presenti; i dati,

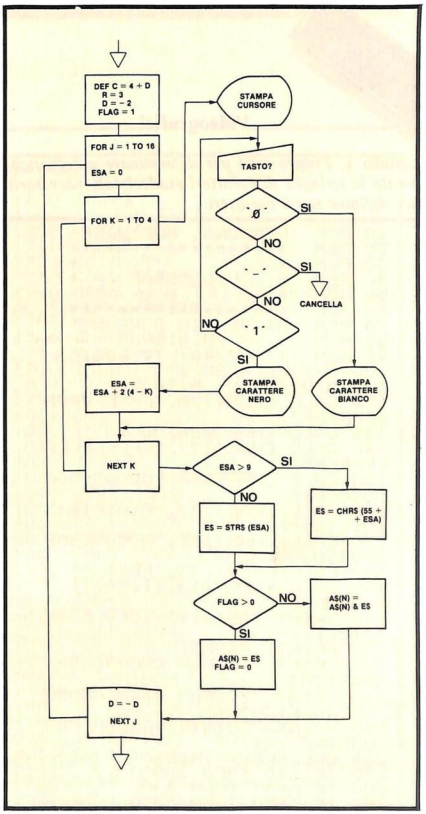


Figura 2. *Routine per generare la stringa di caratteri esadecimali che definiscono un carattere. Il ciclo K esplora blocchi di 4 pixel; se K = 1, ESA = ESA + 2 ^ (4 − K), cioè 8; per K = 2, ESA = 4, ecc.*

comprendenti le coordinate, vengono posti nella matrice B(30,3) e successivamente registrati su nastro. Per procedere alla registrazione, come ho già detto, bisogna premere SHIFT(+) appena terminato il disegno. L'uso di questa matrice si rende necessario poichè, quando si dà la OPEN in output, compaiono le istruzioni per l'uso del registratore e il disegno si sposta verso l'alto fuori campo cosicchè non può più essere riprodotto.

Per richiamare l'immagine è necessario inserire nel nuovo programma la parte finale di questo listato che inizia con OPEN ≠ 10:"CS1",SEQUENTIAL, INPUT, INTERNAL, FIXED (linea 1740) e specificare le coordinate relative per porre il disegno nella posizione opportuna, specificando eventualmente il colore. ■

Listato 1. *Programma per determinare automaticamente le stringhe di caratteri esadecimali, necessarie per definire nuovi caratteri.*

```
 10 REM   PERSONAL SOFTWARE
 20 REM   ******************
 30 REM   *                *
 40 REM   *    VIDEOGRAF   *
 50 REM   *                *
 60 REM   ******************
 70 REM   DI SERGIO BORSANI
 80 REM         28.6.1983
 90 REM   TI-99/4A  TI BASIC
100 OPTION BASE 1
110 DIM A$(30),B(24,3)
120 CALL CHAR(128,"FFFFFFFFF
FFFFFFF")
130 CALL CHAR(126,"0")
140 CALL CHAR(127,"FFFFFFFFF
FFFFFFF")
150 CALL CHAR(130,"000000000
00000FF")
160 CALL CHAR(131,"010101010
1010101")
170 CALL CHAR(132,"808080808
0808080")
180 CALL CHAR(133,"FF")
190 CALL COLOR(12,11,16)
200 CALL CLEAR
210 PRINT TAB(8);"VIDEO GRAF
ICI":::::
220 PRINT "PREMI:"
230 PRINT "1. PER CREARE UN
GRAFICO"
240 PRINT "2. PER RICHIAMARE
 UN GRAFICO"
250 PRINT "FCTN(4)=CLEAR PER
 USCIRE":::::
260 CALL KEY(0,K,S)
270 IF S=0 THEN 260
280 IF K=49 THEN 300
290 IF K=50 THEN 1730
300 CALL CLEAR
310 CALL HCHAR(3,16,130,12)
320 CALL VCHAR(4,15,131,12)
330 CALL VCHAR(4,28,132,12)
340 CALL HCHAR(16,16,133,12)
350 N=0
360 N=N+1
370 IF N<24 THEN 450
380 FINE$="FINE CARATTERI"
390 FOR J=1 TO LEN(FINE$)
400 L=ASC(SEG$(FINE$,J,1))
410 CALL HCHAR(18,15+J,L)
420 NEXT J
430 N=23
440 GOTO 1280
450 CALL HCHAR(20,1,32,16)
460 FOR R=4 TO 11
470 CALL HCHAR(R,3,126,8)
480 NEXT R
490 R=3
500 D=-2
510 DEF C=4+D
520 FLAG=1
530 FOR J=1 TO 16
540 ESA=0
550 FOR K=1 TO 4
560 CALL HCHAR(R+INT((J+1)/2
),C+K,127)
```

Seguito figura 1.

```
570 CALL KEY(0,KEY,S)
580 IF S=0 THEN 570
590 IF KEY=48 THEN 650
600 IF KEY=45 THEN 450
610 IF KEY<>49 THEN 570
620 CALL HCHAR(R+INT((J+1)/2
),C+K,128)
630 ESA=ESA+2^(4-K)
640 GOTO 660
650 CALL HCHAR(R+INT((J+1)/2
),C+K,126)
660 NEXT K
670 IF ESA>9 THEN 700
680 E$=STR$(ESA)
690 GOTO 710
700 E$=CHR$(55+ESA)
710 IF FLAG=0 THEN 750
720 A$(N)=E$
730 FLAG=0
740 GOTO 760
750 A$(N)=A$(N)&E$
760 D=-D
770 NEXT J
780 DOM$="PROCEDO?"
790 FOR J=1 TO LEN(DOM$)
800 L=ASC(SEG$(DOM$,J,1))
810 CALL HCHAR(20,2+J,L)
820 NEXT J
830 CALL KEY(0,KEY,S)
840 IF S=0 THEN 830
850 IF KEY=83 THEN 890
860 IF KEY=78 THEN 450
870 GOTO 830
880 CALL HCHAR(20,1,32,32)
890 NC=136+N
900 RIGA$="CHR$ "&STR$(NC)&"
: "&CHR$(34)&A$(N)&CHR$(34)
910 CALL HCHAR(1,1,32,32)
920 FOR J=1 TO LEN(RIGA$)
930 L=ASC(SEG$(RIGA$,J,1))
940 CALL HCHAR(1,2+J,L)
950 NEXT J
960 X=4
970 Y=22
980 CALL CHAR(NC,A$(N))
990 CALL HCHAR(X,Y,NC)
1000 CALL KEY(0,KEY,S)
1010 IF S=0 THEN 1000
1020 XVECCHIO=X
1030 YVECCHIO=Y
1040 IF KEY=69 THEN 1110
1050 IF KEY=88 THEN 1140
1060 IF KEY=83 THEN 1170
1070 IF KEY=68 THEN 1200
1080 IF KEY=49 THEN 1230
1090 IF KEY=43 THEN 1270
1100 GOTO 1000
1110 IF X=4 THEN 1000
1120 X=X-1
1130 GOTO 1240
1140 IF X=15 THEN 1000
1150 X=X+1
1160 GOTO 1240
1170 IF Y=16 THEN 1000
1180 Y=Y-1
1190 GOTO 1240
1200 IF Y=27 THEN 1000
1210 Y=Y+1
1220 GOTO 1240
1230 GOTO 360
1240 CALL HCHAR(XVECCHIO,YVE
CCHIO,32)
1250 CALL HCHAR(X,Y,NC)
1260 GOTO 1000
1270 REM   *** REGISTRA ***
1280 REG$="VUOI REGISTRARE?
(S/N)"
1290 CALL HCHAR(20,1,32,32)
```

Seguito figura 1.

```
1300 FOR J=1 TO LEN(REG$)
1310 L=ASC(SEG$(REG$,J,1))
1320 CALL HCHAR(20,5+J,L)
1330 NEXT J
1340 CALL KEY(0,KEY,S)
1350 IF S=0 THEN 1340
1360 IF KEY=78 THEN 200
1370 IF KEY=83 THEN 1390
1380 GOTO 1340
1390 RR$="RUNNING"
1400 FOR J=1 TO LEN(RR$)
1410 L=ASC(SEG$(RR$,J,1))
1420 CALL HCHAR(24,11+J,L)
1430 NEXT J
1440 CTR=0
1450 FOR J=1 TO 12
1460 FOR K=1 TO 12
1470 CALL GCHAR(3+J,15+K,CX)
1480 IF CX=32 THEN 1530
1490 CTR=CTR+1
1500 B(CTR,1)=J
1510 B(CTR,2)=K
1520 B(CTR,3)=CX
1530 NEXT K
1540 NEXT J
1550 CALL CLEAR
1560 IF CTR>0 THEN 1610
1570 PRINT "NON C'E' NIENTE!
"
1580 FOR T=1 TO 500
1590 NEXT T
1600 GOTO 110
1610 OPEN #10:"CS1",SEQUENTI
AL,OUTPUT,INTERNAL,FIXED
1620 PRINT #10:N
1630 FOR J=1 TO N
1640 PRINT #10:B(J,1);B(J,2)
;B(J,3)
1650 NEXT J
1660 J=1
1670 PRINT #10:A$(J);A$(J+1)
```

Seguito figura 1.

```
;A$(J+2)
1680 J=J+3
1690 IF J>N THEN 1710
1700 GOTO 1670
1710 CLOSE #10
1720 GOTO 200
1730 CALL CLEAR
1740 OPEN #10:"CS1",SEQUENTI
AL,INPUT ,INTERNAL,FIXED
1750 N=0
1760 INPUT #10:N
1770 FOR J=1 TO N
1780 INPUT #10:B(J,1),B(J,2)
,B(J,3)
1790 NEXT J
1800 J=1
1810 INPUT #10:A$(J),A$(J+1)
,A$(J+2)
1820 J=J+3
1830 IF J>N THEN 1850
1840 GOTO 1810
1850 CLOSE #10
1860 CALL CLEAR
1870 FOR J=1 TO N
1880 CALL CHAR(136+J,A$(J))
1890 NEXT J
1900 PRINT "COORDINATE"::
1910 INPUT "RIGA? ":XO
1920 INPUT "COLONNA? ":YO
1930 CALL CLEAR
1940 FOR J=1 TO N
1950 RIG=XO+B(J,1)
1960 COL=YO+B(J,2)
1970 IF RIG<1 THEN 2020
1980 IF RIG>24 THEN 2020
1990 IF COL<1 THEN 2020
2000 IF COL>32 THEN 2020
2010 CALL HCHAR(RIG,COL,B(J,
3))
2020 NEXT J
2030 GOTO 2030
2040 END
```


TELEMATICA

Dal viewdata all'office automation

di Riccardo Glucksmann
Cod. 518D Pag. 186
L. 19.000

SCONTO 20% AGLI ABBONATI FINO AL 28-2-'84

Tutti oggi parlano di telematica, di società dell'informazione, di banche dati.
Ma cosa è la telematica? Un insieme di servizi di videoinformazione e trasmissione di dati e testi. Innanzitutto la videoinformazione. Essa rappresenta un servizio che, utilizzando le reti telefoniche pubbliche, permette ad un qualsiasi utente, dotato di un televisore a colori adatto, di richiedere e ricevere informazioni memorizzate su opportune banche di dati (Videotel e Televideo). Poi vi sono i servizi pubblici per la trasmissione di testi scritti da terminale a terminale ed il fac-simile. Essi sono basilari, fra l'altro, per la realizzazione della "posta elettronica".
Le applicazioni della telematica sono infinite ed in parte ancora da scoprire. Esso è, innanzitutto, un nuovo e potente "medium" nel campo della comunicazione e dell'informazione, ma è anche lo strumento principale che rivoluzionerà l'organizzazione e la produttività del lavoro di ufficio, per realizzare quello che si chiama "office automation".
Questo libro intende dare un impulso alla conoscenza della telematica, e si prefigge di offrire al lettore un panorama dei problemi connessi con questa disciplina e con i relativi aspetti applicativi. Le caratteristiche dell'esposizione fanno si che il volume possa proporsi indifferentemente all'esperto EDP e di organizzazione, quanto allo studioso che si accosta per la prima volta a questa materia: l'esperto troverà un sicuro riferimento per la risoluzione di problemi teorici e pratici, mentre lo studioso troverà, in una forma organica, i principi fondamentali indispensabili per la conoscenza delle varie problematiche.

Sommario

Telematica e suo sviluppo - Evoluzione delle telecomunicazioni per lo sviluppo della telematica - Reti per telecomunicazioni - Reti di calcolatori e banche dati - Videotex e Teletext - Altri nuovi servizi di telematica - Funzionalità del sistema videotex - Sviluppi del videotex nel mondo - Telematica in Italia - Sviluppo delle comunicazioni - Applicazioni della Telematica - Comunicazioni di massa e aspetti socio-economici e giuridici.

Potete acquistare il suddetto libro nelle migliori librerie oppure scrivendo direttamente a: **Gruppo Editoriale Jackson - Divisione Libri - Via Rosellini, 12 20124 Milano**

Come controllare la memoria del TI99/4A

Il signor Paolo Roncali di Cordenons (PN), propone alcune tecniche per tenere sotto controllo l'occupazione della memoria del TI99/4A ed evitare spiacevoli sorprese.

Il seguente sottoprogramma può essere inserito all'inizio del programma principale oppure, cambiando i numeri di linea, chiamato al momento opportuno da una GO TO ...

```
1 KL = KL + 0.00883
2 PRINT KL
3 GO SUB 1
```

Dopo aver battuto il RUN, vedremo scorrere sullo schermo una continua sequenza di numeri che si fermerà solo alla comparsa del MEMORY FULL.

L'ultima cifra visualizzata corrisponderà appunto ai Kbyte di memoria che restano ancora liberi.

Il tempo impiegato è relativamente lungo: circa 20 secondi per Kbyte.

È possibile comunque diminuire notevolmente i tempi di attesa modificando il sottoprogramma nel seguente modo:

```
1 CALL CLEAR
2 PRINT "DOPO - MEMORY FULL - BATTI:":::
  "PRINT ::: KL        (ENTER)"
3 KL = KL + 0.00883
4 GO SUB 3
```

Una volta raggiunto il MEMORY FULL bisognerà quindi battere PRINT ::: KL senza farlo precedere da un numero di linea; apparirà così un solo numero che corrisponde appunto ai Kbyte liberi.

Vediamo ora come è stato ricavato il numero 0.00883; ciò servirà soprattutto per adattare tale sottoprogramma ad altri tipi di calcolatori o comunque per altre capacità di memoria.

Per prima cosa è necessario eseguire (questa volta da solo) il seguente programma:

```
10 N = N + 1
20 PRINT N
30 GO SUB 10
```

Anche in questo caso vedremo scorrere sullo schermo una continua sequenza di numeri, che si fermerà solo alla comparsa del MEMORY FULL.

Nel caso del TI 99/4A l'ultima cifra è: 1811.

Si divide poi la capacità massima di memoria del computer (16 Kbyte) con l'ultima cifra letta (1811); il risultato di tale divisione è appunto 0.00883 che corrisponde al valore unitario di memoria.

Un'altra breve considerazione riguarda l'ultima istruzione del sotto programma: il GO SUB non può essere sostituito con GO TO!

Il primo sottoprogramma e di conseguenza anche il secondo, potrebbero essere modificati in diversi modi.

Per esempio:

```
1 KL = KL + 0.00883
2 KI = 16 - KL
3 PRINT KI, KL
4 GO SUB 1
```

fornirà sia la quantità di memoria impegnata dal programma principale (KI), sia quella libera (KL). Oppure:

```
1 KL = KL + 0.00883
2 NN = NN + 1
3 IST = 513 - INT(NN/3.5)
4 PRINT KL, IST
5 GO SUB 1
```

in questo caso avremo sia il valore della memoria ancora libera (KL) che il relativo numero massimo di istruzioni *medie* (IST) che sarebbe ancora possibile aggiungere al programma principale.

Naturalmente quest'ultimo valore non può che essere approssimato e deriva dalla considerazione che una media di 32 istruzioni occupa (nel TI99/4A) circa 1 Kbyte, purchè tra queste non compaiano matrici da una o più dimensioni (una sola istruzione del tipo DIM Z(42,42) è sufficiente per avere il MEMORY FULL).

Il numero 513 che troviamo nella terza istruzione corrisponde al massimo numero di istruzioni *medie* possibili e lo si ricava moltiplicando 32 per 16 (Kbyte); mentre il numero 3.5 che figura nella mede-

PERSONAL SOFTWARE

Come controllare la memoria del TI99/4A

sima istruzione è il risultato della divisione tra 1811 (già visto prima) e 513; rappresenta l'unità di istruzione *media* corrispondente all'unità di memoria (0.00883).

Vediamo ora un altro metodo, molto più semplice e soprattutto più rapido, anche se assai approssimato, che permetterà comunque di sapere se è stato raggiunto oppure no un certo limite di memoria.

È sufficiente inserire (o se già esiste, aggiungere) nel programma principale il dimensionamento di una matrice fittizia, per esempio una DIM Z(450).

Avremo in questo modo una specie di lampada spia che indicherà la "riserva" nel "serbatoio" della memoria.

Infatti se dopo aver battuto il RUN non apparirà il messaggio MEMORY FULL avremo la certezza di avere ancora a disposizione oltre 4 Kbyte di memoria; nel caso contrario si può provare a cambiare il valore di dimensionamento della matrice, ad esempio con una DIM Z(228) ed in questo caso, se non comparirà il MEMORY FULL sapremo di avere ancora a disposizione più di 2 Kbyte.

Ritengo che la seguente tabellina sia sufficiente per una stima approssimata della memoria disponibile.

DIM Z(900)	8 Kbyte
DIM Z(450)	4 Kbyte
DIM Z(228)	2 Kbyte
DIM Z(114)	1 Kbyte
DIM Z(57)	0.5 Kbyte

Comunque volendo creare una tabellina più dettagliata si può ricorrere al seguente programma:

```
10 K = K + 0.00883
20 N = N + 1
30 PRINT N, K
40 GO SUB 10
```

Le cifre che si vedranno scorrere a sinistra dello schermo corrispondono al valore di dimensionamento della matrice e quelle a destra ai Kbyte.

Tale programma potrà quindi anche rivelarsi utile per conoscere quanta memoria occupa il dimensionamento di una matrice. ■

EVM
post market

PET-CBM

COMPILAT. DTL 4/8000	L. 330.000
COMP. PC/BASIC 3/4/8	L. 350.000
COMP. PET-SPEED 8000	L. 350.000
ULTRA KRAM	L. 300.000
SUPER BASIC	L. 90.000
DISK REVEALED	L. 60.000
LOCKSMITH	L. 40.000
WORD-CRAFT	L. 280.000
CHIAVI PROT. DTL DA	L. 35.000
EPROM PROGRAMMER PET	L. 270.000
80 COLONNE KIT DA 3032 A 8032	L. 260.000
CAVI DA	L. 50.000
APPLESOFT LOADER	L. 80.000
WORD PROCESS 3/4/8000	L. 180.000
ASSEMBLER 3/4/8000	L. 150.000

VIC-20

40 COLONNE SOFTWARE	L. 45.000
40/80 COLONNE HARDWARE	L. 180.000
VIC EPROM PROGRAMMER	L. 180.000
VIC MON	L. 29.000
HI-RES+3K GRAFICA	L. 54.000
MAGAZZINO	L. 150.000
RAM 3K	L. 49.000
RAM 8K	L. 76.000
RAM 16K	L. 118.000
PROGRAMMERS AID	L. 29.000
4 SLOT MOTHER BOARD	L. 69.000
FATTURAZIONE	L. 150.000

CBM-64

BASIC 4.0 WEDGE	L. 95.000
SPRITE GENERATOR	L. 40.000
MAGAZZINO	L. 150.000
CASSA NEGOZI	L. 150.000
CAVO CBM64-CENTRONICS	L. 38.000
CHARACTERS GENERATORS	L. 30.000
FATTURAZIONE	L. 150.000
MUSIC	L. 40.000

LIBRI

GUIDA AL VIC 20	L. 25.000
PERIFERICHE VIC	L. 9.500
CORSO DI ASSEMBLER CON PROGRAMMI SU NASTRO	L. 35.000
TAVOLE COMMODORE	in preparazione
PERIFERICHE COMMODORE	L. 13.500
GUIDA AL CBM 64	L. 25.000
IL BASIC 4.0 COMMODORE	L. 15.000
IL S.O. DEL 64 DISASS.	L. 38.000
CATALOGO EVM	gratis

E.V.M. SOFTWARE BANK

Vi dà la possibilità di accedere e contribuire con pochissima spesa ad una vasta biblioteca di programmi. SCRIVETE PER INFORMAZIONI!!!!

Per ricevere il catalogo EVM POST MARKET su prodotti COMMODORE, SOFTWARE, PRODOTTI DI CONSUMO, COMPONENTI, STAMPANTI, ecc. **SCIVETECI!!!!**

SCONTI AI SIGNORI RIVENDITORI.

Condizioni di vendita:

I prezzi con la sola esclusione dei libri sono IVA esclusa. Franco Montevarchi. Pagamento Contrassegno.

IL NOSTRO INDIRIZZO È:
E.V.M. POST MARKET
VIA MARCONI 9/A
52025 MONTEVARCHI (AR)
tel. (055) 982513

NOME .....
INDIRIZZO .....
C.A.P ..... **CITTÀ** .....
INVIATEMI
☐ **CATALOGO**
☐ **INFORMAZIONI SU E.V.M. SOFTWARE BANK**
☐ .....

PERSONAL SOFTWARE

Grafici ad alta risoluzione con il TI99/4A

Un programma BASIC che vi permette di visualizzare qualunque funzione matematica

di *Sergio Borsani*

È proprio vero che l'erba del vicino è sempre più verde! Penso al disagio che prova qualcuno nel constatare che il proprio computer non è in grado di offrire alcune prestazioni desiderate. So di qualcuno che pensava di poter ottenere con un TI-99 grafici tridimensionali sul tipo di quelli rivolti alla progettazione industriale. Scagli la prima pietra chi non ha sperato di poter ottenere almeno il grafico ad alta risoluzione di una funzione matematica. La difficoltà consiste nel fatto che in TI BASIC non è possibile indirizzare indipendentemente i punti video (pixel). Tuttavia si può, grazie al nuovo video processor TMS9918A che il TI99/4A possiede al posto del TMS9918 del precedente modello, definire un carattere punto per punto e successivamente stamparlo nella posizione opportuna.

Con l'home computer, un programma BASIC che abbia la pretesa di costruire un grafico ad alta risoluzione deve innanzitutto "scandire" il video da sinistra a destra pixel per pixel e calcolare, in corrispondenza delle 256 colonne, il valore della funzione che si vuole rappresentare.

Per distinguere le 24 righe e le 32 colonne disponibili nel modo grafico dalle 192 righe e 256 colonne disponibili nel modo bit-mat, potremmo convenire di chiamare queste ultime pix-righe e pix-colonne.

Al termine della prima operazione il computer avrà memorizzato, per ogni pix-colonna, il numero di pix-riga corrispondente al valore della funzione. Dovrà poi colmare tutte le lacune eventualmente esistenti tra ogni punto ed il successivo, mettendo in ON tutti i pixel tra essi compresi.

Ogniqualvolta si verifica il salto di riga o di colonna passerà quindi a definire la stringa esadecimale atta a specificare il carattere particolare che rappresenta un piccolo tratto della curva considerata e a stamparlo nella posizione opportuna con l'istruzione CALL HCHAR.

Questo per quanto riguarda i concetti di base.

Il programma che presento nel listato 1 ovviamente è strutturato in conformità a quanto ho premesso e in più possiede delle prerogative che saranno sicuramente molto apprezzate dai lettori. Per citare le più importanti, una consiste nel dimensionamento automatico del grafico, un'altra nell'uso di una routine per la "somma" dei caratteri in modo che la curva non cancelli parzialmente gli assi di riferimento quando li interseca o anche quando passa in prossimità di essi.

Quest'ultima soluzione semplice ed originale è il naturale complemento di un programma rivolto alla grafica ad alta risoluzione e merita una breve descrizione.

Sommiamo i caratteri

Se si stampa un carattere sul video esso sostituisce ogni altro simbolo eventualmente presente nella stessa posizione; alla regola non sfuggono nemmeno i caratteri speciali.

Se un tratto di curva passa in prossimità di un asse cartesiano o addirittura lo interseca, i caratteri che lo rappresentano andranno a sostituire quelli che rappresentavano l'asse creando dei vuoti a dir poco antiestetici. La "somma" dei caratteri consiste nel definire, quando è necessario, nuovi caratteri che portano in ON tutti i pixel del carattere che rappresenta un tratto di curva e di quello che invece rappresenta una parte dell'asse di riferimento.

Per far questo le stringhe esadecimali che definiscono i caratteri vengono decodificati ed i loro simboli vengono sommati ad uno ad uno in base ad una tabella di composizione memorizzata nel programma (istruzioni DATA delle linee 300-440).

Ogni simbolo esadecimale controlla un blocco di 4 pixel; F si riferisce ad un blocco con tutti e quattro i pixel in posizione ON. È evidente

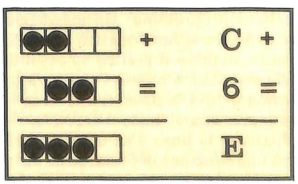


Figura 1. *È possibile sommare due caratteri sommando uno ad uno gli elementi delle stringhe esadecimali che li definiscono in base ad una tabella di composizione.*

Grafici ad alta risoluzione con il TI99/4A

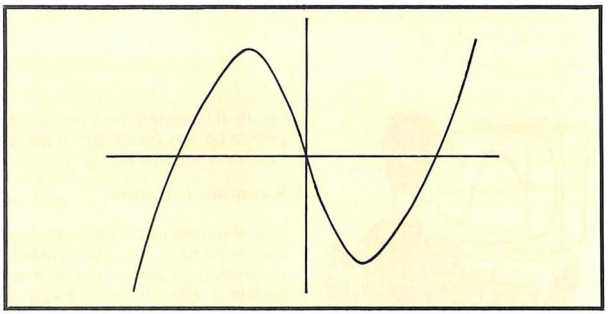


Figura 2. *A titolo dimostrativo nel programma è memorizzata la funzione* $y = x^3 - 3x$. *È qui riprodotta l'immagine elaborata dal computer.*

che F sommato a qualsiasi altro simbolo esadecimale dà ancora F; per la costanza dei risultati questo elemento non è stato inserito nella tabella.

Consideriamo ad esempio la somma C + 6, il risultato è E; infatti C è un blocco con il primo ed il secondo pixel ON ed il terzo ed il quarto OFF, 6 è un blocco con il secondo ed il terzo pixel ON; la loro somma è un blocco con il primo, il secondo ed il terzo pixel ON ed il quarto OFF, esso corrisponde al simbolo esadecimale E.

La figura 1 illustra questo concetto di somma.

Utilizziamo il programma

Prima di esaminare dettagliatamente il listato vediamo come si utilizza il programma. Un messaggio ricorda che la funzione della quale si vuole ottenere il grafico va definita alla riga 850. Chi non l'avesse fatto prima del RUN deve fermare il programma con il tasto FCTN(4), richiamare la linea 850, dove troverà già una funzione definita, sostituirla con la propria e lanciare nuovamente il programma. Proseguendo, bisogna specificare l'intervallo nel quale si studia la funzione, X1 è l'estremo inferiore e X2 è l'estremo superiore.

Il grafico è dimensionato automaticamente nel senso che sia la posizione degli assi sia l'unità di misura vengono calcolate per ottenere la migliore rappresentazione, in modo da utilizzare tutti i 192 pixel sull'asse verticale evitando che una parte della curva debba cadere fuori dallo schermo.

L'unica attenzione da parte dell'utente deve essere volta ad escludere dall'intervallo valori per i quali non può essere calcolata la funzione.

Ad esempio la funzione Y = SQR (1 − X x X) è definita solo nell'intervallo − 1 + 1; per valori diversi il radicando diventa negativo, appare il messaggio BAD ARGUMENT ed il programma si blocca. Se la funzione è Y = 36/X, l'intervallo non può comprendere il valore 0 perchè non esiste il valore della funzione in quel punto; ad esempio, si potrà ottenere il grafico nell'intervallo + 1 + 36. In questo caso l'asse delle Y non appare sullo schermo ed anche il grafico appare tutto spostato sulla sinistra. Prevedendo tali situazioni, il programma è stato dotato di una routine che consente il cambiamento dell'origine degli assi cartesiani, controllato dai quattro tasti muniti di frecce.

Riferendoci ancora alla funzione Y = 36/X, nell'intervallo + 1 + 36, premendo un tasto al termine del dimensionamento appare il solo asse orizzontale nell'ultima riga in basso, l'asse delle Y è virtualmente spostato a sinistra fuori dallo schermo.

Premendo due o tre volte il tasto (D) con la freccia rivolta a destra è possibile ottenere la stampa anche dell'asse Y.

È opportuno non abusare di questa funzione e spostare gli assi solo del minimo necessario perchè così si sposta anche tutta la stampa del grafico ed una parte di questo potrà non apparire sul video.

La figura 2 mostra il grafico della funzione Y = X^ 3−3 x X, attualmente presente nel programma, nell'intervallo − 2 + 2, senza traslazione degli assi.

Il computer impiega poco più di due minuti per il dimensionamento automatico e circa quattro minuti per la stampa del grafico con l'utilizzazione di 74 caratteri definiti da programma.

Un tuffo nel listato

Alla riga 230 vengono dimensionate le matrici; S$ contiene la tabella di composizione per la somma dei caratteri, P è il numero di pixcolonna dei punti del diagramma calcolati per valori discreti, ESA$ memorizza gli elementi per comporre la stringa esadecimale che definisce i caratteri grafici, infine E$ contiene in sequenza i simboli esadecimali.

Delle istruzioni DATA si è già parlato in precedenza, esse contengono la tabella di composizione per la somma dei caratteri.

Alla linea 680 e seguente i dati in input: X1 ed X2 sono gli estremi dell'intervallo nel quale viene studiata la funzione. Alla riga 850 c'è la funzione della quale si vuole ottenere il grafico.

Le istruzioni 920-1070 ricavano il valore unitario corrispondente ad un pixel; esso è posto inizialmente

Grafici ad alta risoluzione con il TI99/4A

uguale alla duecentoquarantesima parte dell'intervallo X2-X1, se per tale valore il massimo relativo (MAX) o il minimo relativo (MIN) cadono fuori dallo schermo, allora il valore unitario (U) è posto uguale alla centosettantaseiesima parte dell'intervallo MAX-MIN. In tal modo tutta la funzione, da Y = f (X1) a Y = f (X2), viene rappresentata sullo schermo.

RIGA 0 e COLO 0 (righe 1110-1120) sono le coordinate dell'origine mentre X0 e Y0 sono le stesse espresse in numero di pix-colonna e pix-riga.

Il ciclo FOR NEXT (righe 1160-1280) carica i valori della matrice unidimensionale P (vedi figura 3).

INIZIO (1250) e FINE (1290) indicano proprio l'inizio e la fine della stampa, in numero di pix-colonna.

Alla riga 1340 viene mandato l'ultimo messaggio, in seguito saranno ridefiniti tutti i caratteri ASCII che non saranno più disponibili per il testo.

Segue la stampa degli assi cartesiani, TRSX0 e TRSY0 controllano la traslazione degli assi, il loro valore è determinato nel sottoprogramma 2920.

Con il ciclo FOR NEXT 1820 inizia la stampa del grafico.

Le principali variabili utilizzate sono P1, numero di pix-riga del primo punto, RIGA 1 e COLO 1, numeri di riga e colonna, NRC, numero di pix-riga del pixel all'interno del carattere, NCC, numero di pix-colonna del punto all'interno del carattere (un carattere è formato da 8 pix-righe e 8 pix-colonne), NB1, numero del blocco nel quale si trova il pixel (un carattere è formato da 16 blocchi di 4 pixel ciascuno), ESADEC1 è il corrispondente valore decimale del simbolo esadecimale al quale si risale con la funzione E$ (ESADEC1). Al punto seguente, P2, sono associate le stesse variabili con indice 2.

Al termine di ogni ciclo il vecchio punto conseguente diventa il nuovo antecedente ed un nuovo punto diventa il nuovo conseguente (istruzioni 2270-2300).

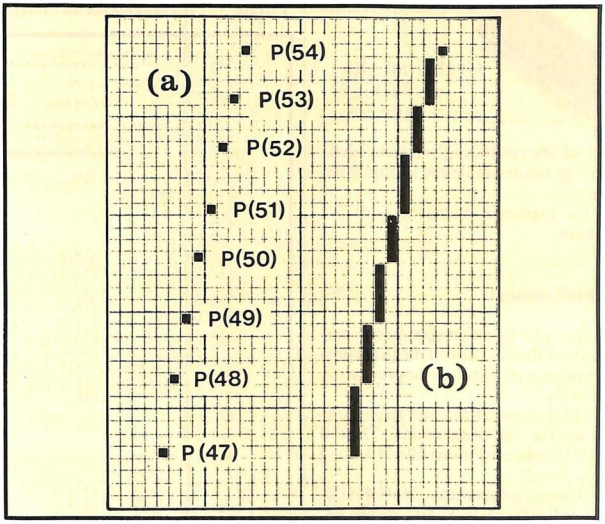


Figura 3. *Nella fase di dimensionamento si calcola la curva per punti (a). Successivamente vengono colmati i vuoti tra punto e punto e definiti i caratteri per la stampa. La mappa-video (b) mostra, ingrandito, un breve tratto della curva in esame.*

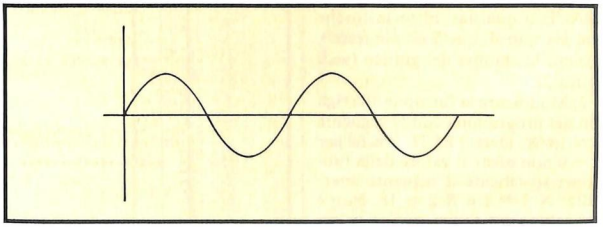


Figura 4. *Dimensionamento automatico e traslazione degli assi consentono sempre la migliore rappresentazione grafica. La funzione è Y = SIN(X).*

Se tra due punti P1 e P2 la differenza tra i numeri di pix-riga è maggiore di 1 il ciclo FOR NEXT 2000-2090 si incarica di stampare punti sulla stessa pix-colonna di P1 fino a colmare il vuoto esistente tra i due punti (la differenza tra i numeri di pix-colonna, invece, è sempre uguale a 1).

Anche durante questo ciclo se cambia il numero di riga o di colonna il carattere deve essere stampato.

Quando P1 e P2 si trovano sulla stessa pix-riga il programma controlla se si trovano sullo stesso blocco tra i 16 che formano un carattere perchè se ciò si verifica va calcolato il nuovo valore esadecimale da associare al blocco (linea 2240).

Nell'ultima parte del listato le REM rendono esplicito il contenuto delle quattro routine utilizzate dal programma.

La funzione CALL GCHAR (linea 2500) indaga se lo spazio destinato alla stampa di un carattere è già occupato, nel qual caso si procede alla somma dei caratteri.

La stampa di un carattere è sempre seguita dall'azzeramento di

PERSONAL SOFTWARE

Grafici ad alta risoluzione con il TI99/4A

ESA$ che contiene gli elementi della stringa esadecimale che lo hanno definito.

La memoria richiesta è di 7.1 Kbyte.

Ultimi consigli

Qualche breve suggerimento permetterà di utilizzare al meglio il programma e di acquisire la pratica necessaria.

1) Lasciate la funzione attualmente scritta nel programma. Date il RUN. Indicate l'intervallo X 1 = −2 e X 2 = 2.

Apparirà la scritta DIMENSIONAMENTO RUNNING. Dopo poco più di due minuti appaiono alcuni messaggi. Premendo un tasto qualsiasi si ottiene la stampa degli assi cartesiani con l'origine al centro del teleschermo. Premendo il tasto SPACE o qualsiasi altro tasto che non sia uno di quelli con le frecce, inizierà la stampa del grafico (vedi figura 2).

2) Modificare la funzione alla riga 850 del programma con la seguente Y = 18/X. Date il RUN. Poichè per X = 0 non esiste il valore della funzione, specificate il seguente intervallo: X 1 = 1 e X 2 = 18. Non è opportuno attribuire a X 1 valori prossimi allo zero perchè per X tendente a zero il valore della funzione sale rapidamente e, per rappresentare tutto il tratto di curva, viene notevolmente ridotta, in proporzione, la rappresentazione dell'intervallo di definizione.

Dopo l'ultimo messaggio, premendo un tasto non appare nulla sullo schermo: l'origine degli assi cade al di fuori. Si potrebbe ottenere ugualmente il grafico della funzione: con la stampa degli assi esso è tuttavia più completo.

Premete allora il tasto E (↑), apparirà l'asse orizzontale, poi tre volte il tasto D (→) e apparirà anche l'asse verticale. Premete un altro tasto qualsiasi e verrà stampato il grafico.

3) Inserite la funzione Y = 3 x X + 4SQR (1 − X x X). Abbiamo già detto che il radicando non può essere negativo, pertanto X non può essere minore di − 1 nè maggiore di + 1. Indicate l'intervallo X 1 = − 1 e X 2 = 1. Procedete come in prima.

Non è necessaria la traslazione degli assi perciò dopo la loro stampa premete un tasto qualsiasi eccettuati quelli con le frecce.

4) Inserite la funzione Y = SIN(X) ed indicate l'intervallo X 1 = 0 e X 2 = 12.6. I valori di X sono espressi in radianti e 12.6 rappresenta circa 4 π. L'origine degli assi apparirà spostata a sinistra piuttosto in alto. Il computer ha calcolato esattamente lo spazio necessario per la stampa del grafico, tuttavia questa avverrà nella parte superiore dello schermo. Se desiderate centrare l'immagine azionate il tasto X (↓) più volte fino a portare l'origine nella posizione più opportuna, quindi premete un tasto qualsiasi per la stampa (vedi figura 4). ■

Listato 1. *Programma per la stampa di grafici in alta risoluzione.*

```
100 REM  PERSONAL SOFTWARE
110 REM  ******************
120 REM  *                *
130 REM  * GRAFICI AD ALTA *
140 REM  *                *
150 REM  *   RISOLUZIONE  *
160 REM  *                *
170 REM  ******************
180 REM  di Sergio Borsani
190 REM  Tel.:(0332)73026
200 REM  versione: TI BASIC
210 CALL CLEAR
220 PRINT "  GRAFICO DI UNA FUNZIONE"::::::::::::
230 DIM S$(14,14),P(240),ESA$(16),E$(15)
240 RESTORE 300
250 FOR J=0 TO 14
260 FOR K=0 TO 14
270 READ S$(J,K)
280 NEXT K
290 NEXT J
300 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E
310 DATA 1,1,3,3,5,5,7,7,9,9,B,B,D,D,F
320 DATA 2,3,2,3,6,7,6,7,A,B,A,B,E,F,E
330 DATA 3,3,3,3,7,7,7,7,B,B,B,B,F,F,F
340 DATA 4,5,6,7,4,5,6,7,C,D,E,F,C,D,E
350 DATA 5,5,7,7,5,5,7,7,D,D,F,F,D,D,F
360 DATA 6,7,6,7,6,7,6,7,E,F,E,F,E,F,E
370 DATA 7,7,7,7,7,7,7,7,F,F,F,F,F,F,F
380 DATA 8,9,A,B,C,D,E,F,8,9,A,B,C,D,E
390 DATA 9,9,B,B,D,D,F,F,9,9,B,B,D,D,F
400 DATA A,B,A,B,E,F,E,F,A,B,A,B,E,F,E
410 DATA B,B,B,B,F,F,F,F,B,B,B,B,F,F,F
420 DATA C,D,E,F,C,D,E,F,C,D,E,F,C,D,E
430 DATA D,D,F,F,D,D,F,F,D,D,F,F,D,D,F
440 DATA E,F,E,F,E,F,E,F,E,F,E,F,E,F,E
450 REM
460 REM    CARATTERI PER
470 REM    DISEGNARE GLI
480 REM    ASSI
490 REM
500 Z$(1)="01010101010101"
510 Z$(2)="FF00000000000000"
520 Z$(3)="FF01010101010101"
530 CALL CHAR(33,Z$(1))
540 CALL CHAR(34,Z$(2))
550 CALL CHAR(35,Z$(3))
560 REM
570 REM        ISTRUZIONI
580 REM
590 CALL CLEAR
600 PRINT "  DEFINIRE LA FUNZIONE ALLA"
610 PRINT "LINEA 850 DEL PROGRAMMA":::
620 PRINT "  SE LO AVETE GIA' FATTO "
630 PRINT "SCRIVETE GLI ESTREMI"
640 PRINT "DELL'INTERVALLO DI"
650 PRINT "DEFINIZIONE (X1<X2)."::
660 PRINT "  IL GRAFICO E' DIMENSIONATO"
670 PRINT "AUTOMATICAMENTE":::
680 INPUT "X1 = ":X1
690 INPUT "X2 = ":X2
700 IF X1<X2 THEN 740
710 PRINT
720 PRINT "DEVE ESSERE X1<X2, RIPROVA":::
730 GOTO 680
740 TRSX=0
750 TRSY=0
760 TRSX0=0
770 TRSY0=0
780 CALL CLEAR
790 PRINT TAB(7);"DIMENSIONAMENTO":::::
800 PRINT TAB(11);"RUNNING"::::::
810 REM
820 REM  ******************
830 REM
840 REM
850 DEF Y=X^3-3*X
860 REM
870 REM
880 REM       FUNZIONE
890 REM
900 REM  ******************
910 REM
920 UX=(X2-X1)/240
930 X=X1
940 MAX=Y
950 MIN=Y
960 FOR I=1 TO 240
970 X=X1+I*UX
980 IF Y<=MAX THEN 1000
990 MAX=Y
```

PERSONAL SOFTWARE

Grafici ad alta risoluzione con il TI99/4A

Seguito listato 1.

```
1000 IF Y>=MIN THEN 1020
1010 MIN=Y
1020 NEXT I
1030 UY=(MAX-MIN)/176
1040 IF UX>UY THEN 1070
1050 U=UY
1060 GOTO 1080
1070 U=UX
1080 IF U=0 THEN 2920
1090 X1O=-X1/UX
1100 Y1O=MAX/U
1110 COLO0=INT(X1O/8)+1
1120 RIGA0=INT(Y1O/8)+1
1130 Y0=COLO0*8
1140 Y0=RIGA0*8+1
1150 INIZIO=0
1160 FOR I=1 TO 240
1170 X=(I-X0+7)*U
1180 IF (X>=X1)*(X<=X2)THEN 1210
1190 P(I)=0
1200 GOTO 1220
1210 P(I)=INT(Y0-Y/U)
1220 IF P(I)<1 THEN 1270
1230 IF P(I)>192 THEN 1270
1240 IF INIZIO<>0 THEN 1280
1250 INIZIO=I
1260 GOTO 1280
1270 IF INIZIO<>0 THEN 1290
1280 NEXT I
1290 FINE=I
1300 REM
1310 REM     ASSI CARTESIANI
1320 REM
1330 CALL CLEAR
1340 PRINT "     ESEGUITO IL GRAFICO."
1350 PRINT "   PER USCIRE DAL PROGRAMMA"
1360 PRINT "PREMERE IL TASTO (Q), OPPURE"
1370 PRINT TAB(10);"FCTN(4),":::
1380 PRINT " PRIMA DELLA STAMPA SI PUO'"
1390 PRINT "   CAMBIARE L'ORIGINE DEGLI"
1400 PRINT " ASSI CARTESIANI PREMENDO I"
1410 PRINT "     TASTI CON LE FRECCE.":::
1420 PRINT "     PREMERE UN TASTO PER"
1430 PRINT TAB(10);"CONTINUARE.":::::
1440 CALL KEY(0,KEY,STATUS)
1450 IF STATUS=0 THEN 1440
1460 CALL CLEAR
1470 COLD0=COLO0+TRSX0
1480 RIGA0=RIGA0+TRSY0
1490 RIGA01=RIGA0+1
1500 IF (RIGA01<1)+(RIGA01>24)THEN 1540
1510 CALL HCHAR(RIGA01,2,34,30)
1520 FLAG=0
1530 GOTO 1550
1540 FLAG=1
1550 IF (COLO0<1)+(COLO0>32)THEN 1590
1560 CALL VCHAR(2,COLO0,33,22)
1570 FLAG=0
1580 GOTO 1600
1590 FLAG=1
1600 IF FLAG=1 THEN 1620
1610 CALL HCHAR(RIGA01,COLO0,35)
1620 GOSUB 2920
1630 IF TRASLA=1 THEN 1460
1640 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
1650 RESTORE 1640
1660 FOR J=0 TO 15
1670 READ E$(J)
1680 NEXT J
1690 NUMCAR=35
1700 GOSUB 2850
1710 P1=P(INIZIO)
1720 RIGA1=INT((P1-1)/8)+1
1730 COL01=INT((INIZIO-1)/8)+1
1740 NRC=P1-INT((P1-1)/8)*8
1750 NCC=INIZIO-INT((INIZIO-1)/8)*8
1760 NB1=NRC*2
1770 IF NCC>4 THEN 1800
1780 NB1=NB1-1
1790 GOTO 1810
1800 NCC=NCC-4
1810 ESADEC1=2^(4-NCC)
1820 FOR PIX=INIZIO+1 TO FINE-1
1830 P2=P(PIX)
1840 RIGA2=INT((P2-1)/8)+1
1850 COL02=INT((PIX-1)/8)+1
1860 NRC=P2-INT((P2-1)/8)*8
1870 NCC=PIX-INT((PIX-1)/8)*8
1880 NB2=NRC*2
1890 IF NCC>4 THEN 1920
1900 NB2=NB2-1
1910 GOTO 1930
1920 NCC=NCC-4
1930 ESADEC2=2^(4-NCC)
1940 IF P1=P2 THEN 2150
1950 ESA$(NB1)=E$(ESADEC1)
1960 DIFF=P2-P1
1970 W=SGN(DIFF)
1980 VOLTE=ABS(DIFF)-1
1990 IF VOLTE=0 THEN 2100
2000 FOR Z=1 TO VOLTE
2010 NB1=NB1+W*2
2020 IF (NB1<1)+(NB1>16)THEN 2040
2030 GOTO 2080
```

Seguito listato 1.

```
2040 GOSUB 2400
2050 GOSUB 2850
2060 RIGA1=RIGA1+W
2070 NB1=NB1-W*16
2080 ESA$(NB1)=E$(ESADEC1)
2090 NEXT Z
2100 IF RIGA1<>RIGA2 THEN 2120
2110 IF COL01=COL02 THEN 2260
2120 GOSUB 2400
2130 GOSUB 2850
2140 GOTO 2250
2150 REM  *** STESSA PIX-RIGA ***
2160 IF COL01=COL02 THEN 2210
2170 ESA$(NB1)=E$(ESADEC1)
2180 GOSUB 2400
2190 GOSUB 2850
2200 GOTO 2260
2210 IF NB1=NB2 THEN 2240
2220 ESA$(NB1)=E$(ESADEC1)
2230 GOTO 2260
2240 ESADEC1=ESADEC1+ESADEC2
2250 GOTO 2270
2260 ESADEC1=ESADEC2
2270 P1=P2
2280 RIGA1=RIGA2
2290 COL01=COL02
2300 NB1=NB2
2310 NEXT PIX
2320 CALL KEY(0,KEY,STATUS)
2330 IF STATUS=0 THEN 2320
2340 IF KEY=81 THEN 3200
2350 IF KEY=113 THEN 3200
2360 GOTO 2320
2370 REM
2380 REM       STAMPA
2390 REM
2400 CARAT$=ESA$(1)
2410 FOR I=2 TO 16
2420 CARAT$=CARAT$&ESA$(I)
2430 NEXT I
2440 NUMCAR=NUMCAR+1
2450 IF NUMCAR>159 THEN 2320
2460 RIGA=RIGA1+TRSY
2470 IF (RIGA<1)+(RIGA>24)THEN 2550
2480 COLONNA=COL01+1+TRSX
2490 IF (COLONNA<1)+(COLONNA>32)THEN 2550
2500 CALL GCHAR(RIGA,COLONNA,CC)
2510 IF CC=0 THEN 2530
2520 GOSUB 2590
2530 CALL CHAR(NUMCAR,CARAT$)
2540 CALL HCHAR(RIGA,COLONNA,NUMCAR)
2550 RETURN
2560 REM
2570 REM    SOMMA CARATTERI
2580 REM
2590 IF CC<33 THEN 2800
2600 IF CC>35 THEN 2800
2610 C$=" "
2620 FOR I=1 TO 16
2630 L1=ASC(SEG$(CARAT$,I,1))
2640 L2=ASC(SEG$(Z$(CC-32),1,1))
2650 IF L1>60 THEN 2680
2660 L1=L1-48
2670 GOTO 2690
2680 L1=L1-55
2690 IF L2>60 THEN 2720
2700 L2=L2-48
2710 GOTO 2730
2720 L2=L2-55
2730 IF L1=15 THEN 2770
2740 IF L2=15 THEN 2770
2750 C$=C$&S$(L1,L2)
2760 GOTO 2780
2770 C$=C$&"F"
2780 NEXT I
2790 CARAT$=SEG$(C$,2,16)
2800 RETURN
2810 REM
2820 REM      AZZERAMENTO
2830 REM         ESA$
2840 REM
2850 FOR I=1 TO 16
2860 ESA$(I)="0"
2870 NEXT I
2880 RETURN
2890 REM
2900 REM     TRASLAZIONE
2910 REM
2920 CALL KEY(0,TASTO,ST)
2930 IF ST=0 THEN 2920
2940 IF TASTO=69 THEN 3000
2950 IF TASTO=88 THEN 3040
2960 IF TASTO=83 THEN 3080
2970 IF TASTO=68 THEN 3120
2980 TRASLA=0
2990 GOTO 3160
3000 TRSY=TRSY-1
3010 TRSY0=-1
3020 TRSX0=0
3030 GOTO 3150
3040 TRSY=TRSY+1
3050 TRSY0=1
3060 TRSX0=0
3070 GOTO 3150
3080 TRSX=TRSX-1
3090 TRSY0=0
3100 TRSX0=-1
3110 GOTO 3150
3120 TRSX=TRSX+1
3130 TRSY0=0
3140 TRSX0=1
3150 TRASLA=1
3160 RETURN
3170 REM
3180 REM      FINE
3190 REM
3200 CALL CLEAR
3210 END
```

TEXAS TI99/4A

Il movimento del TI BASIC

di Sergio Borsani

Il TI99/4A, come è noto, può essere programmato in TI BASIC, Extended BASIC, Assembler, Pascal e Logo che, verso la fine di quest'anno, uscirà nella versione italiana. Per i giochi o i programmi didattici che richiedono un intenso movimento il più adatto è l'Assembler che più si avvicina al linguaggio macchina ma che tuttavia è difficile da usare. I primi due linguaggi elencati all'inizio invece sono di gran lunga i più usati e si differenziano notevolmente per quanto riguarda la capacità di creare oggetti in movimento. L'Extended BASIC infatti è provvisto di sottoprogrammi, richiamabili con istruzioni CALL..., rivolti espressamente alla grafica e fornisce così la possibilità di creare un ambiente molto simile a quello dei più comuni videogames.

Mentre ho intenzione di illustrare questo argomento in un prossimo numero, desidero ora soffermarmi sul TI BASIC.

Questo linguaggio, residente nella console, è poco adatto alle animazioni grafiche e per utilizzarlo a questo scopo è necessario seguire alcune regole: (a), gli oggetti che si muovono devono essere piccoli, preferibilmente formati da un solo carattere; (b), il loro numero deve essere limitato; (c), i cicli che sono adibiti al movimento devono contenere solo le istruzioni essenziali, tutte le altre dovranno essere poste all'interno di essi.

Il movimento ovviamente si crea stampando uno space, CHR$ (32), nella vecchia posizione e collocando lo stesso oggetto in una posizione nuova. L'effetto è inferiore se si invertono le istruzioni cioè se prima si stampa il carattere nella nuova posizione e poi si cancella nella vecchia. Il movimento non può essere uniforme ma apparirà a scatti poichè l'oggetto balza da una riga o da una colonna ad un'altra. Per ottenere una maggiore velocità, i salti potranno comprendere più di una riga o di una colonna a scapito della già precaria uniformità del movimento.

Se l'oggetto è formato da più di un carattere esso si sposterà parte per parte e l'andatura assumerà un aspetto ondeggiante poichè il più delle volte alcuni caratteri che formano l'oggetto occuperanno già la nuova posizione mentre altri dovranno ancora essere trasferiti.

Anche se gli oggetti sono piccoli e formati da un solo carattere, ricordiamo che un programma TI BASIC può spostarli soltanto uno alla volta. È evidente che se si volesse creare un gioco come quello del calcio, dove è richiesto il movimento della palla e di molti giocatori, per spostare la linea d'attacco il programma dovrebbe compiere numerosi cicli per le proprie mosse e quelle dell'avversario rendendo l'esecuzione di una lentezza esasperante. Una complicazione deriva anche dal fatto che un carattere cancella un altro sul quale viene stampato, pertanto il ciclo che crea il movimento deve contenere le istruzioni perchè i caratteri non si sovrappongano o per ricrearli dopo un'eventuale sovrapposizione.

```
100 CALL CLEAR
110 CALL CHAR(128,"F")
120 CALL COLOR(13,2,2)
130 X1=16
140 Y1=12
150 CALL HCHAR(Y1,X1,128)
160 CALL KEY(0,K,S)
170 IF S=0 THEN 160
180 IF K=69 THEN 230
190 IF K=88 THEN 260
200 IF K=83 THEN 290
210 IF K=68 THEN 320
220 GOTO 160
230 Y2=Y1-1
240 X2=X1
250 IF Y2<1 THEN 160 ELSE 350
260 Y2=Y1+1
270 X2=X1
280 IF Y2>24 THEN 160 ELSE 350
290 X2=X1-1
300 Y2=Y1
310 IF X2<1 THEN 160 ELSE 350
320 X2=X1+1
330 Y2=Y1
340 IF X2>32 THEN 160
350 CALL HCHAR(Y1,X1,32)
360 CALL HCHAR(Y2,X2,128)
370 X1=X2
380 Y1=Y2
390 GOTO 160
```

Listato 1. *Un semplice esempio di animazione.*

Se si desidera controllare il movimento da tastiera bisogna inserire l'istruzione CALL KEY. Solitamente si associano i quattro tasti con le frecce ai movimenti nelle quattro direzioni; una tipica sequenza di istruzioni è quella del listato 1.

In questo esempio, premendo i quattro tasti con le frecce, un quadratino nero si sposta in tutte le direzioni.

È importante controllare il valore delle coordinate ed impedire la stampa quando il numero di riga è minore di 1 o maggiore di 24 e quando il numero di colonna è minore di 1 o maggiore di 32.

Il listato 2 è un programma didattico sulle frazioni. Si tratta di tagliare da un rettangolo una parte frazionaria controllando da tastiera un paio di forbici che appaiono sullo schermo. Il tasto (S) le sposta a sinistra, il tasto (D) a destra ed il tasto (T) provoca il movimento automatico che simula il taglio. Per ogni prova sono consentiti due tentativi, dopo due errori consecutivi il computer suggerisce la risposta esatta.

Al termine dei quesiti viene indicato il livello raggiunto.

La tecnica di programmazione è conforme ai concetti su esposti. Una complicazione è dovuta al fatto che le forbici, formate da due o quattro caratteri, partono dall'esterno del rettangolo colorato, vi si addentrano lasciando alle spalle la traccia del taglio e poi escono dalla parte opposta passando per due volte dallo sfondo di un colore a quello di un altro.

Dopo aver utilizzato il programma, apparirà evidente come le animazioni grafiche siano importanti non solo per i giochi, ma trovino anche una loro precisa collocazione nei programmi educativi. ■

Listato 2. *Esempio di animazione applicato alla didattica.*

```
100 REM  PERSONAL SOFTWARE
110 REM  ******************
120 REM  *                *
130 REM  *    FRAZIONI    *
140 REM  *                *
150 REM  ******************
160 REM  di: Sergio Borsani
170 REM  versione: TI BASIC
180 CALL CLEAR
190 PRINT TAB(11);"FRAZIONI":::::::::
200 CALL CHAR(128,"0")
210 CALL CHAR(129,"0101010101010101")
220 A$="00101B1B0C0C0607"
230 B$="0301031E32263C18"
240 C$="0103030303030303"
250 D$="0301070D09090F06"
260 CALL CHAR(130,A$)
270 CALL CHAR(131,B$)
280 CALL CHAR(132,C$)
290 CALL CHAR(133,D$)
300 CALL CHAR(136,A$)
310 CALL CHAR(137,"00103030606C0C0")
320 CALL CHAR(138,B$)
330 CALL CHAR(139,"800080F098C87830")
340 CALL CHAR(140,C$)
350 CALL CHAR(141,"0080808080808080")
360 CALL CHAR(142,D$)
370 CALL CHAR(143,"8000C0602020E0C0")
380 CALL CHAR(144,"00000000FF")
390 CALL CHAR(145,"FF8181")
400 CALL CHAR(146,"FF0101")
410 CALL COLOR(13,2,14)
420 CALL CLEAR
430 PRINT "HAI A DISPOSIZIONE UN PAIO"
440 PRINT "DI FORBICI PER TAGLIARE PAR-"
450 PRINT "TI FRAZIONARIE"::::
460 PRINT "USA I SEGUENTI COMANDI:";::
470 PRINT "TASTO (D) PER SPOSTARE LE"
480 PRINT TAB(11);"FORBICI A DESTRA"
490 PRINT "TASTO (S) PER SPOSTARE LE"
500 PRINT TAB(11);"FORBICI A SINISTRA"
510 PRINT "TASTO (T) PER TAGLIARE.":::::
520 PRINT TAB(8);"premi un tasto"
530 PRINT TAB(8);"per continuare"
540 CALL KEY(0,K,S)
550 IF S=0 THEN 540
560 PUNTI=0
570 NUME=1
580 DENO=2
590 GOSUB 1100
600 NUME=1
610 DENO=4
620 GOSUB 1100
630 NUME=1
640 DENO=3
650 GOSUB 1100
660 NUME=3
670 DENO=4
680 GOSUB 1100
690 NUME=2
```

Seguito listato 2.

```
700 DENO=3
710 GOSUB 1100
720 NUME=5
730 DENO=6
740 GOSUB 1100
750 NUME=5
760 DENO=12
770 GOSUB 1100
780 NUME=2
790 DENO=4
800 GOSUB 1100
810 NUME=3
820 DENO=6
830 GOSUB 1100
840 NUME=7
850 DENO=12
860 GOSUB 1100
870 CALL CLEAR
880 IF PUNTI<20 THEN 920
890 PRINT TAB(11);"OTTIMO!":::::::::
900 GOSUB 2730
910 GOTO 1030
920 IF PUNTI<15 THEN 960
930 PRINT TAB(12);"BENE!":::::::::
940 GOSUB 2730
950 GOTO 1030
960 IF PUNTI<10 THEN 1000
970 PRINT TAB(9);"SUFFICIENTE.":::::::::
980 GOSUB 2500
990 GOTO 1030
1000 PRINT "TEMO CHE LA MATEMATICA NON"
1010 PRINT "SIA IL TUO FORTE.":::::::::
1020 GOSUB 2630
1030 PRINT "VUOI CONTINUARE? (S/N)"
1040 CALL KEY(0,K,S)
1050 IF S=0 THEN 1040
1060 IF (K=83)+(K=115)THEN 560
1070 IF (K=78)+(K=110)THEN 2850
1080 GOTO 1040
1090 REM  *** ANIMAZIONE ***
1100 CALL CLEAR
1110 FOR J=3 TO 11
1120 CALL HCHAR(J,11,128,12)
1130 NEXT J
1140 F$="TAGLIA"
1150 FOR J=1 TO 6
1160 L=ASC(SEG$(F$,J,1))
1170 CALL HCHAR(17,10+J,L)
1180 NEXT J
1190 LN=LEN(STR$(NUME))
1200 LD=LEN(STR$(DENO))
1210 FOR J=1 TO LN
1220 L=ASC(SEG$(STR$(NUME),J,1))
1230 CALL HCHAR(16,19-LN+J,L)
1240 NEXT J
1250 CALL HCHAR(17,19,144)
1260 IF (LN=1)*(LD=1)THEN 1280
1270 CALL HCHAR(17,18,144)
1280 FOR J=1 TO LD
1290 L=ASC(SEG$(STR$(DENO),J,1))
1300 CALL HCHAR(18,19-LD+J,L)
1310 NEXT J
1320 XVECCHIO=11
1330 CALL HCHAR(13,XVECCHIO,136)
1340 CALL HCHAR(13,XVECCHIO+1,137)
1350 CALL HCHAR(14,XVECCHIO,138)
1360 CALL HCHAR(14,XVECCHIO+1,139)
1370 CALL KEY(0,K,S)
1380 IF S=0 THEN 1370
1390 IF (K=68)+(K=100)THEN 1440
1400 IF (K=83)+(K=115)THEN 1470
1410 IF (K=84)+(K=116)THEN 1600
1420 GOTO 1370
1430 REM  *** DESTRA ***
1440 XDIR=1
1450 GOTO 1480
1460 REM  *** SINISTRA ***
1470 XDIR=-1
1480 XN=XVECCHIO+XDIR
1490 IF (XN<11)+(XN>21)THEN 1370
1500 XNUOVO=XN
```

Il movimento del TI BASIC

Seguito listato 2.

```
1510 CALL HCHAR(13,XVECCHIO,32,2)
1520 CALL HCHAR(14,XVECCHIO,32,2)
1530 CALL HCHAR(13,XNUOVO,136)
1540 CALL HCHAR(13,XNUOVO+1,137)
1550 CALL HCHAR(14,XNUOVO,138)
1560 CALL HCHAR(14,XNUOVO+1,139)
1570 XVECCHIO=XNUOVO
1580 GOTO 1370
1590 REM  *** TAGLIA ***
1600 X1=XVECCHIO
1610 X2=XVECCHIO+1
1620 CALL HCHAR(13,X1,32,2)
1630 CALL HCHAR(14,X1,32,2)
1640 CALL HCHAR(12,X1,136)
1650 CALL HCHAR(12,X2,137)
1660 CALL HCHAR(13,X1,138)
1670 CALL HCHAR(13,X2,139)
1680 CALL HCHAR(11,X1,130)
1690 CALL HCHAR(12,X1,138)
1700 CALL HCHAR(12,X2,139)
1710 CALL HCHAR(13,X1,32,2)
1720 CALL HCHAR(11,X1,132)
1730 CALL HCHAR(12,X1,142)
1740 CALL HCHAR(12,X2,143)
1750 CALL HCHAR(11,X1,130)
1760 CALL HCHAR(12,X1,138)
1770 CALL HCHAR(12,X2,139)
1780 CALL HCHAR(10,X1,130)
1790 CALL HCHAR(11,X1,131)
1800 CALL HCHAR(12,X1,32,2)
1810 FOR R=10 TO 4 STEP -1
1820 CALL HCHAR(R,X1,132)
1830 CALL HCHAR(R+1,X1,133)
1840 CALL HCHAR(R,X1,130)
1850 CALL HCHAR(R+1,X1,131)
1860 CALL HCHAR(R-1,X1,130)
1870 CALL HCHAR(R,X1,131)
1880 CALL HCHAR(R+1,X1,129)
1890 NEXT R
1900 CALL HCHAR(3,X1,132)
1910 CALL HCHAR(4,X1,133)
1920 CALL HCHAR(3,X1,130)
1930 CALL HCHAR(4,X1,131)
1940 CALL HCHAR(2,X1,136)
1950 CALL HCHAR(2,X2,137)
1960 CALL HCHAR(3,X1,131)
1970 CALL HCHAR(4,X1,129)
1980 CALL HCHAR(1,X1,136)
1990 CALL HCHAR(1,X2,137)
2000 CALL HCHAR(2,X1,138)
2010 CALL HCHAR(2,X2,139)
2020 CALL HCHAR(3,X1,129)
2030 FOR T=1 TO 100
2040 NEXT T
2050 COLO=10+12*NUME/DENO
2060 IF X1=COLO THEN 2380
2070 IF VOLTE=2 THEN 2170
2080 E$="ERRORE! RIPROVA"
2090 FOR J=1 TO 15
2100 L=ASC(SEG$(E$,J,1))
2110 CALL HCHAR(21,8+J,L)
2120 NEXT J
2130 FOR T=1 TO 1000
2140 NEXT T
2150 VOLTE=2
2160 GOTO 1100
2170 VOLTE=0
2180 EE$="ERRORE!"
2190 FOR J=1 TO 7
2200 L=ASC(SEG$(EE$,J,1))
2210 CALL HCHAR(21,10+J,L)
2220 NEXT J
2230 GOSUB 2630
2240 RR$="GUARDA LA RISPOSTA ESATTA."
2250 FOR J=1 TO 26
2260 L=ASC(SEG$(RR$,J,1))
2270 CALL HCHAR(23,2+J,L)
2280 NEXT J
2290 CALL HCHAR(12,11,145)
2300 CALL HCHAR(12,12,146,11)
2310 CALL HCHAR(13,COLO,136)
```

Seguito listato 2.

```
2320 CALL HCHAR(13,COLO+1,137)
2330 CALL HCHAR(14,COLO,138)
2340 CALL HCHAR(14,COLO+1,139)
2350 FOR T=1 TO 3000
2360 NEXT T
2370 GOTO 2490
2380 ES$="ESATTO!"
2390 FOR J=1 TO 7
2400 L=ASC(SEG$(ES$,J,1))
2410 CALL HCHAR(21,10+J,L)
2420 NEXT J
2430 GOSUB 2500
2440 IF VOLTE=2 THEN 2470
2450 PUNTI=PUNTI+2
2460 GOTO 2480
2470 PUNTI=PUNTI+1
2480 VOLTE=0
2490 RETURN
2500 CALL SOUND(400,440,8)
2510 CALL SOUND(200,392,8)
2520 CALL SOUND(200,440,7)
2530 CALL SOUND(400,587,6)
2540 CALL SOUND(200,523,5)
2550 CALL SOUND(200,587,4)
2560 CALL SOUND(400,494,3)
2570 CALL SOUND(200,440,4)
2580 CALL SOUND(200,494,5)
2590 CALL SOUND(500,392,6)
2600 FOR T=1 TO 300
2610 NEXT T
2620 RETURN
2630 CALL SOUND(400,440,6)
2640 CALL SOUND(150,440,6)
2650 CALL SOUND(400,349,6)
2660 CALL SOUND(150,330,6)
2670 CALL SOUND(400,175,6)
2680 CALL SOUND(150,165,6)
2690 CALL SOUND(1000,131,6)
2700 FOR T=1 TO 800
2710 NEXT T
2720 RETURN
2730 CALL SOUND(300,698,2)
2740 CALL SOUND(300,932,2)
2750 CALL SOUND(300,784,2)
2760 CALL SOUND(300,880,2)
2770 CALL SOUND(150,932,2)
2780 CALL SOUND(150,880,2)
2790 CALL SOUND(150,784,2)
2800 CALL SOUND(150,880,2)
2810 CALL SOUND(500,698,2)
2820 FOR T=1 TO 300
2830 NEXT T
2840 RETURN
2850 CALL CLEAR
2860 END
```

DEBUG

Dal listato BASIC del Programmer's tool kit, pubblicato nel n. 12/13 è saltata una linea fondamentale per il corretto funzionamento del programma. Occorre aggiungere la seguente istruzione:
1305 DATA A9, 2C, 20, D2, FF, A5, FE.

TI99/4A

Il movimento in TI Extended BASIC

di Sergio Borsani

Ricordo con simpatia le parole dell'amico di Milano: "mi tuffo nell'Assembler e non riemergerò che tra qualche mese". Condivido il disappunto di quanti vorrebbero eseguire in BASIC programmi dove la rapidità dell'azione e la molteplicità degli oggetti in movimento giocano un ruolo determinante ma credo che non si debbano sottovalutare le istruzioni grafiche relative agli sprite fornite dal Modulo TI Extended BASIC. Semmai io sento una limitazione nell'impossibilità di poter usare, dal BASIC, il video in Bit-Map e nella mancanza di un comando CALL LINE per tracciare linee.

Evidentemente queste diverse esigenze sono dovute al fatto che generalmente scrivo programmi didattici e per me la velocità non è essenziale; ma anche chi preferisce dedicarsi ai video game potrà ottenere con l'Extended BASIC risultati soddisfacenti senza dover ricorrere, come con altri computer, alle istruzioni PEEK e POKE per accedere direttamente alle locazioni di memoria o, addirittura, "tuffarsi" nell'Assembler.

Le subroutine alle quali mi riferisco sono SPRITE, MOTION, LOCATE, POSITION, DISTANCE, COINC e DELSPRITE. Le prerogative più evidenti sono il poter determinare la posizione di un oggetto in pixel, senza essere vincolati alle 24 righe e 32 colonne, ed il movimento automatico degli sprite.

Per creare il movimento in Extended BASIC si può procedere essenzialmente in due modi: variare successivamente la posizione con l'istruzione CALL LOCATE o variare la velocità con il comando CALL MOTION. Nel primo caso il moto risulterà poco uniforme, cioè a scatti, soprattutto a velocità elevate; in compenso risulterà molto preciso in quanto si ha continuamente sotto controllo la posizione dello sprite. Nel secondo caso si otterrà un movimento più uniforme ma facilmente si perderà il controllo dello sprite. La ragione di ciò è che, dal momento in cui si determinano le nuove componenti della velocità fino a qundo viene data l'istruzione CALL MOTION, lo sprite continua a spostarsi automaticamente lungo la vecchia direzione; pertanto la variazione di velocità giunge sempre con un lieve ritardo rispetto al punto previsto.

Per un confronto tra i due metodi digitate i brevi listati 1 e 2. In entrambi si vuole ottenere una traiettoria circolare ed apparentemente le differenze sono trascurabili, ma provate, nel secondo, a modificare la linea 240 con V = − 30; non solo è aumentata la velocità ma la traiettoria si è trasformata da circolare a elittica. Per V = − 40 lo sprite, addirittura, sfugge al controllo ed esce dal video.

Per ottenere una traiettoria circolare avremmo dovuto cambiare la velocità ad ogni istante; noi purtroppo siamo costretti ad operare ad intervalli discreti. Il listato 3 è un esempio, se vogliamo banale, di come il movimento possa essere controllato con l'uso degli joystick.

Per un uso corretto non deve essere premuto il tasto ALPHA LOCK. Il programma simula il gioco del rincorrersi: un omino deve raggiungere l'avversario che cerca di sottrarsi e quando ciò avviene compare la scritta "PRESO!". L'istruzione CALL COINC, alla linea 210, controlla se c'è la coincidenza tra i due sprite. In caso affermativo la variabile CO assume il valore − 1, in caso contrario la stessa variabile assume il valore 0.

La posizione dello sprite si identifica con quella del pixel in alto a sinistra della piccola superficie quadrata che contiene lo sprite stesso. Se nell'istruzione CALL COINC si dà un valore non nullo alla tolleranza, si può ottenere la coincidenza anche quando gli sprite sono solo parzialmente sovrapposti, senza che coincidano i punti caratteristici che individuano le loro posizioni. Nel nostro esempio la tolleranza è stata posta uguale a 20 ed è una buona norma che essa sia tanto maggiore quanto più grandi sono gli sprite e quanto maggiore è la loro velocità.

Un automobilista che debba sterzare improvvisamente davanti ad un ostacolo potrà farlo con successo solo se i tempi di reazione lo consentono. Quando il programma prevede il controllo di un oggetto in movimento bisogna tenere conto dei tempi di reazione dell'interprete BASIC e con la pratica e l'esperienza si otterranno le migliori soluzioni. Il listato 4 è un esempio di come il BASIC debba prendere d'anticipo le istruzione che riguardano il movimento e come questo possa essere controllato con opportune pause tra una istruzione e l'altra.

Una pallina aggira un ostacolo che si trova lungo la sua traiettoria e l'ostacolo è posto in corrispondenza della colonna 18. Teoricamente la deviazione della pallina dovrebbe avvenire quando questa raggiunge la colonna 128 (in pixel), in realtà, alla linea 210 del programma, si controlla se è stata superata la colonna 120; quegli 8 pixel di scarto permettono al BASIC di eseguire l'istruzione successiva per tempo, senza che la pallina passi sull'ostacolo.

Le istruzioni CALL MOTION alle linee 220-250 sono state separate da cicli FOR-NEXT il cui solo scopo è quello di creare una pausa durante la quale il movimento prosegue automaticamente. Se la veloci-

Il movimento in TI Extended BASIC

tà è uguale a 20, per T che varia da 1 a 100, lo sprite percorre una distanza pari a circa 26 pixel mentre, per T che varia da 1 a 200, lo sprite percorre una distanza di 50 pixel. Come si vede la proporzionalità non è rigorosa.

Oltre a questi brevi esempi voglio presentare un gioco completo che, come al solito, riunisce in modo sintetico tutte le considerazioni precedenti (listato 5).

Partecipano due concorrenti ognuno dei quali "aiuta" una gallina ad attraversare un'autostrada. Il concorrente di sinistra ha a disposizione i tasti (1) e (2) rispettivamente per far avanzare la gallina e per fermarla; analogamente il giocatore di destra dispone dei tasti (9) e (0). Si guadagna un punto per ogni gallina che riesce ad attraversare l'autostrada. Si noti come nello schermo siano presenti contemporaneamente 20 sprite e come si ottenga con sufficiente precisione il controllo delle concidenze per determinare se una gallina sia stata investita. ■

```
100 REM MOVIMENTO CIRCOLARE
110 REM ******************
120 REM      STATEMENT
130 REM     CALL LOCATE
140 REM ******************
150 CALL CLEAR
160 CALL CHAR(128,"3C7EFFFFFFFF7E3C")
170 CALL CHAR(129,"0101010101010107")
180 CALL CHAR(130,"00000000000000FF")
190 CALL CHAR(131,"0101010101010FF")
200 CALL CHAR(132,"0101")
210 CALL COLOR(13,9,1)
220 CALL HCHAR(12,1,130,32):: CALL HCHAR(13,1,132,32)
230 CALL VCHAR(1,16,129,24):: CALL HCHAR(12,16,131)
240 YO=124 :: V0=92 :: RAGGIO=60
250 CALL SPRITE(#1,128,2,Y0,X0+RAGGIO
260 ALFA=0
270 ALFA=ALFA+PI/36
280 X=X0+RAGGIO*COS(ALFA):: Y=Y0-RAGGIO*SIN(ALFA)
290 CALL LOCATE(#1,Y,X):: GOTO 270
```

Listato 1. *Traiettoria circolare ottenuta mediante la subroutine LOCATE.*

```
100 REM MOVIMENTO CIRCOLARE
110 REM ******************
120 REM      STATEMENT
130 REM     CALL MOTION
140 REM ******************
150 CALL CLEAR
160 CALL CHAR(128,"3C7EFFFFFFFF7E3C")
170 CALL CHAR(129,"0101010101010107")
180 CALL CHAR(130,"00000000000000FF")
190 CALL CHAR(131,"0101010101010FF")
200 CALL CHAR(132,"0101")
210 CALL COLOR(13,9,1)
220 CALL HCHAR(12,1,130,32):: CALL HCHAR(13,1,132,32)
230 CALL VCHAR(1,16,129,24):: CALL HCHAR(12,16,131)
240 V=-20
250 CALL SPRITE(#1,128,2,92,184)
260 ALFA=0
270 ALFA=ALFA+PI/36
280 VX=V*SIN(ALFA):: VY=V*COS(ALFA)
290 CALL MOTION(#1,VY,VX):: GOTO 270
```

Listato 2. *Traiettoria circolare ottenuta mediante la subroutine MOTION.*

```
100 REM   PROVA JOYSTICK
110 REM ******************
120 CALL CLEAR
130 CALL CHAR(128,"0103030307F7F6767670
73F3F303070B0D0C3C383FFFFE0E0E0E0E0E0FCF
C04")
140 CALL SPRITE(#1,128,2,100,50)
150 CALL SPRITE(#2,128,9,100,200)
160 CALL MAGNIFY(3)
170 CALL JOYST(1,X,Y)
180 CALL MOTION(#1,-10*Y,10*X)
190 CALL JOYST(2,X,Y)
200 CALL MOTION(#2,-10*Y,10*X)
210 CALL COINC(#1,#2,20,CO)
220 IF CO=0 THEN 170
230 CALL DELSPRITE(ALL)
240 DISPLAY AT(12,12)BEEP:"PRESO!"
250 FOR TEMPO=1 TO 1000 :: NEXT TEMPO ::
 GOTO 120
```

Listato 3. *Piccolo gioco per verificare la coincidenza tra due sprite. È necessario il joystick.*

```
100 REM     MOVIMENTO
110 REM ******************
120 REM      STATEMENT
130 REM    CALL POSITION
140 REM ******************
150 CALL CLEAR
160 CALL CHAR(128,"3C7EFFFFFFFF7E3C")
170 CALL CHAR(129,"F")
180 CALL COLOR(13,9,9)
190 CALL VCHAR(10,18,129,6)
200 CALL SPRITE(#1,128,2,96,40,0,30)
210 CALL POSITION(#1,Y,X):: IF X<120 THEN 210
220 CALL MOTION(#1,-20,0):: FOR T=1 TO 150 :: NEXT T
230 CALL MOTION(#1,0,20):: FOR T=1 TO 100 :: NEXT T
240 CALL MOTION(#1,20,0):: FOR T=1 TO 150 :: NEXT T
250 CALL MOTION(#1,0,-30)
260 CALL POSITION(#1,Y,X):: IF X<250 THEN 260
270 CALL LOCATE(#1,96,1):: GOTO 210
```

Listato 4. *Un esempio di movimento con aggiornamento dell'ostacolo.*

Listato 5. *Nel gioco delle galline in autostrada sono state applicate le tecniche descritte.*

```
10 REM ******************
20 REM *                *
30 REM *  LE GALLINE    *
40 REM * IN AUTOSTRADA  *
50 REM *                *
60 REM ******************
70 REM  SERGIO BORSANI
80 REM    (0436/3036)
90 REM TI-99/4A EXT.BASIC
100 CALL CLEAR
110 CALL CHAR(96,"7FC1CDCDCDCDC17F"&RPT$
("0",16)&"3F20EFEFEFEF203F"&RPT$("0",16)
)
120 CALL CHAR(100,"FF00FFFFFFFF00FF"&RPT
$("0",16)&"FF01FDFDFDFD01FF"&RPT$("0",16
))
130 CALL CHAR(104,"FF80BFBFBFBF80FF"&RPT
```

Seguito listato 5.

```
$("0",16)&"FF00FFFFFFFF00FF"&RPT$("0",16
))
140 CALL CHAR(108,"FC04F7F7F7F704FC"&RPT
$("0",16)&"FE83B3B3B3B383FE"&RPT$("0",16
))
150 CALL CHAR(112,"007F717171717F00"&RPT
$("0",16)&"00FE16E6E616FE00"&RPT$("0",16
))
160 CALL CHAR(116,"007F686F6F687F00"&RPT
$("0",16)&"00FE8E8E8E8EFE00"&RPT$("0",16
))
170 CALL CHAR(132,"60E1437F7E3C1828"&RPT
$("0",48))
180 CALL CHAR(121,"0000FFFFFFFF")
190 CALL CHAR(122,"0000FF0000FF")
200 CALL CHAR(123,"0000007E")
210 CALL CHAR(128,"F")
220 CALL COLOR(12,16,15)
230 CALL COLOR(13,15,15)
240 FOR RIGA=5 TO 21
250 CALL HCHAR(RIGA,1,128,32)
260 NEXT RIGA
270 CALL HCHAR(5,1,121,32)
280 CALL HCHAR(7,1,123,32)
290 CALL HCHAR(9,1,123,32)
300 CALL HCHAR(11,1,123,32)
310 CALL HCHAR(13,1,122,32)
320 CALL HCHAR(15,1,123,32)
330 CALL HCHAR(17,1,123,32)
340 CALL HCHAR(19,1,123,32)
350 CALL HCHAR(21,1,121,32)
360 CALL SPRITE(#1,132,16,169,72)
370 CALL SPRITE(#2,132,16,169,160)
380 CALL SPRITE(#3,96,9,41,150,0,-8)
390 CALL SPRITE(#4,100,9,41,164,0,-8)
400 FOR A=5 TO 7
410 CALL SPRITE(#A,112,A,57,2^A,0,-10)
420 NEXT A
430 CALL SPRITE(#8,96,13,73,40,0,-12)
440 CALL SPRITE(#9,100,13,73,53,0,-12)
450 CALL SPRITE(#10,112,7,89,50,0,-20)
460 CALL SPRITE(#11,112,11,89,100,0,-20)
470 CALL SPRITE(#12,116,5,105,10,0,18)
480 CALL SPRITE(#13,116,8,105,80,0,18)
490 CALL SPRITE(#14,104,9,121,60,0,12)
500 CALL SPRITE(#15,108,9,121,75,0,12)
510 FOR A=16 TO 18
520 CALL SPRITE(#A,116,A-8+INT(A*A/100),
137,A*(A-12)*2,0,10)
530 NEXT A
540 CALL SPRITE(#19,104,7,153,120,0,8)
550 CALL SPRITE(#20,108,7,153,135,0,8)
560 CALL MAGNIFY(3)
570 DISPLAY AT(3,10)SIZE(3):0
580 DISPLAY AT(3,15)SIZE(3):0
590 CALL KEY(1,K,S)
600 IF S=0 THEN 630
610 IF K=19 THEN CALL MOTION(#1,-5,0)
620 IF K=7 THEN CALL MOTION(#1,0,0)
630 CALL KEY(2,K1,S1)
640 IF S1=0 THEN 670
650 IF K1=9 THEN CALL MOTION(#2,-5,0)
660 IF K1=10 THEN CALL MOTION(#2,0,0)
670 CALL POSITION(#1,X1,Y1)
680 IF X1>100 THEN 820
690 IF X1>68 THEN 780
700 IF X1>36 THEN 740
710 P1=P1+1
720 DISPLAY AT(3,10)BEEP SIZE(3):P1
730 CALL LOCATE(#1,180,72):: GOTO 950
740 FOR A=7 TO 3 STEP -1
750 CALL COINC(#1,#A,8,C)
760 IF C=-1 THEN 930
770 NEXT A :: GOTO 950
780 FOR A=11 TO 8 STEP -1
790 CALL COINC(#1,#A,8,C)
800 IF C=-1 THEN 930
810 NEXT A :: GOTO 950
820 IF X1>168 THEN 950
830 IF X1>132 THEN 880
840 FOR A=15 TO 12 STEP -1
850 CALL COINC(#1,#A,8,C)
860 IF C=-1 THEN 930
870 NEXT A :: GOTO 950
880 FOR A=20 TO 16 STEP -1
890 CALL COINC(#1,#A,8,C)
900 IF C=-1 THEN 930
910 NEXT A
920 GOTO 950
930 CALL SOUND(100,-3,2)
940 CALL LOCATE(#1,169,72)
950 CALL POSITION(#2,X2,Y2)
960 IF X2>100 THEN 1100
970 IF X2>68 THEN 1060
980 IF X2>36 THEN 1020
990 P2=P2+1
1000 DISPLAY AT(3,15)BEEP SIZE(3):P2
1010 CALL LOCATE(#2,180,160):: GOTO 590
1020 FOR A=7 TO 3 STEP -1
1030 CALL COINC(#2,#A,8,D)
1040 IF D=-1 THEN 1200
1050 NEXT A :: GOTO 590
1060 FOR A=11 TO 8 STEP -1
1070 CALL COINC(#2,#A,8,D)
1080 IF D=-1 THEN 1200
1090 NEXT A :: GOTO 590
1100 IF X2>168 THEN 590
1110 IF X2>132 THEN 1160
1120 FOR A=15 TO 12 STEP -1
1130 CALL COINC(#2,#A,8,D)
1140 IF D=-1 THEN 1200
1150 NEXT A :: GOTO 590
1160 FOR A=20 TO 16 STEP -1
1170 CALL COINC(#2,#A,8,D)
1180 IF D=-1 THEN 1200
1190 NEXT A :: GOTO 590
1200 CALL SOUND(100,-3,2)
1210 CALL LOCATE(#2,169,160)
1220 GOTO 590
```

Ne segue ancora una volta un ERROR 8; è tuttavia interessante notare che se n. è maggiore di 255 o minore di 0 si verifica un ERROR 3, equivalente a "number out of range".
Anche ROM, sebbene possa apparire assurdo, ha una sua sintassi: se digitiamo infatti ROM otterremo il solito ERROR 8. (Forse che il computer sia andato a cercare se c'era una memoria ROM interfacciata?).

Possiamo quindi concludere che queste istruzioni sono state forse immesse in previsione di future interfacce o espansioni di memoria, tuttavia tutto è molto vago e dubbioso e sono convinto che ora che ho dato uno spunto, molti di voi troveranno altre cose curiose e impensabili in proposito.

TEXAS TI 99/4A

Print Using in TI BASIC

di Sergio Borsani

Si è già parlato altre volte dei sostanziali vantaggi offerti dal Modulo SSS TI Extended BASIC. Se, tuttavia, molti comandi aggiuntivi sono insostituibili, soprattutto quelli per la creazione ed il controllo degli sprite, altri sono facilmente simulabili con la versione del BASIC residente. Ricordo che in molte riviste, i primi articoli sul TI 99/4A suggerivano un "trucco" per ottenere le funzioni AND e OR usando rispettivamente i comandi IF (A = B) ★ (A = C) THEN ... e IF (A = B + (A = C) THEN ..., dove naturalmente le espressioni tra parentesi sono scritte a puro titolo di esempio. Come queste, altre istruzioni non disponibili in TI BASIC si possono ottenere ugualmente anche se spesso non in modo così semplice come negli esempi sopra citati ma ricorrendo a delle routine più o meno complesse.
L'istruzione Print Using appartiene a questa categoria. La parola chiave Using viene usata per dare un formato alla stampa, sia su video che su stampante, e risulta particolarmente utile quando si debbano tabulare dei dati numerici. Infatti, se si vogliono incolonnare dei numeri con la funzione TAB, essi verranno giustapposti con il margine sinistro e non con il rispetto della posizione delle unità, delle decine e così via.

Se tutti i numeri sono interi, nella stampa, si dovrà rispettare il margine destro. L'istruzione Print Using assolve questo compito in modo molto semplice. Consideriamo innanzitutto la sua sintassi.

```
PRINT USING "######": N
```

In questo caso il numero individuato dalla variabile N viene stampato in un'area riservata di sei caratteri, con la cifra delle unità nell'ultima posizione a destra (le stringhe alfanumeriche vengono invece stampate a partire da sinistra). Se il numero N è composto da più di sei cifre, l'area di stampa riservata viene riempita con sei asterischi ad indicare che il campo non è sufficientemente ampio per contenere il valore indicato. L'istruzione TAB non è disponibile con la Print Using, pertanto chi volesse usarle entrambe dovrà ricorrere a due istruzioni separate consecutive:

```
PRINT TAB(10);:: PRINT USING "######":
                 123
```

Per ottenere gli stessi risultati in TI BASIC la cosa è abbastanza agevole. È necessario trasformare la variabile numerica in variabile di stringa con la funzione STR$, calcolarne la lunghezza con la funzione LEN ed infine stampare il numero a partire da una colonna prefissata meno la lunghezza calcolata.

A parole può sembrare un po' complicato ma è necessaria una sola istruzione:

```
PRINT TAB (10-LEN(STR$(N)));N
```

ed il numero verrà scritto con la cifra delle unità sulla colonna 10, indipendentemente dalla sua lunghezza. Considerate il listato 1. Sono stati memorizzati dieci numeri nell'istruzione DATA di linea 140 ed il ciclo FOR NEXT seguente li stampa due volte: a sinistra con la semplice istruzione PRINT TAB(4);N; e a destra con la variante un po' più complicata appena suggerita.

L'istruzione Print Using permette anche di definire un formato di stampa comprendente il punto decimale. Ad esempio, si può scrivere:

```
PRINT USING "#####.####":N
```

Se si incolonnano in tal modo più numeri decimali, non verranno stampati con il margine a destra, ma il riferimento sarà dato proprio dal punto decimale. Se, in questo caso, un numero possiede più di quattro cifre decimali, verranno rappresentate solo le prime quattro, con un arrotondamento dell'ultima.
123.456789 risulterà: 123.4568. Se invece è la parte intera del numero ad eccedere le posizioni specificate compariranno gli asterischi. Disponendo del solo TI BASIC, se si volessero incolonnare numeri decimali usando l'espressione prima suggerita, si otterrebbe la giusta posizione del margine destro ma, al variare delle cifre decimali, varierebbe la posizione del punto decimale. In questo caso, per simulare l'opzione Using, non è più sufficiente una sola istruzione TI

PERSONAL SOFTWARE

Print Using in TI BASIC

BASIC ma è necessario ricorrere ad una brevissima routine che è riportata nel listato 2. Si deve ricavare la parte intera con la funzione INT e stamparla come si è visto in precedenza; poi, se esiste, si scrive la parte decimale. Inoltre i numeri vanno trasformati in stringhe altrimenti tra la parte intera e quella decimale verranno lasciati degli spazi vuoti.

```
100 REM       SIMULAZIONE
110 REM       PRINT USING
120 REM   *****************
130 CALL CLEAR
140 DATA 450,1620,135,24,142,463,2670,85
0,125,12
150 PRINT "  SCRITTURA     SIMULAZIONE"
160 PRINT "   NORMALE      PRINT USING"
170 PRINT
180 CALL HCHAR(23,3,95,28)
190 FOR J=1 TO 10
200 READ N
210 PRINT TAB(4);N;
220 PRINT TAB(22-LEN(STR$(N)));N
230 NEXT J
240 PRINT :::
250 PRINT TAB(7);"<BATTI UN TASTO>"
260 CALL KEY(0,K,S)
270 IF S=0 THEN 260
280 CALL CLEAR
290 END
```

Listato 1. *L'istruzione 220 del listato simula il comando Print Using permettendo di allineare i numeri con il margine destro indipendentemente da quante sono le cifre che li compongono.*

```
100 REM       SIMULAZIONE
110 REM       PRINT USING
120 REM   *****************
130 CALL CLEAR
140 DATA 42.6,162.24,1350,24.145,1.42
150 DATA 2380.2,45.628,365,42.64,2.58
160 PRINT "  SCRITTURA     SIMULAZIONE"
170 PRINT "   NORMALE      PRINT USING"
180 PRINT
190 CALL HCHAR(23,3,95,28)
200 FOR J=1 TO 10
210 READ N
220 PRINT TAB(4);N;
230 IN=INT(N)
240 PRINT TAB(22-LEN(STR$(IN)));STR$(IN);
250 IF N=IN THEN 270
260 PRINT STR$(N-IN);
270 PRINT
280 NEXT J
290 PRINT :::
300 PRINT TAB(7);"<BATTI UN TASTO>"
310 CALL KEY(0,K,S)
320 IF S=0 THEN 310
330 CALL CLEAR
340 END
```

Listato 2. *Se i numeri contengono una parte decimale, l'istruzione precedente non garantisce più il giusto incolonnamento.*
Si rende allora necessaria la breve routine presente nelle linee 230-270. In questo modo si simula l'istruzione PRINT USING "###.##":N e si ottiene l'incolonnamento con riferimento al punto decimale.

In tutto sono necessarie cinque linee di programma. Ricordo al termine che l'istruzione Print Using si può usare anche con la stampante nel formato:

PRINT #4,USING "####.##":N

ma di questo parlerò più diffusamente in occasione della presentazione di un programma.

LEGGETE VIDEO GIOCHI

LA PRIMA RIVISTA
DI VIDEOGAMES - COMPUTER - GIOCHI ELETTRONICI

PERSONAL SOFTWARE

Indagine su alcune strane figure

Siete sicuri di saper rappresentare il grafico di una funzione?

di *Giulio Morpurgo*

Introduzione

Scopo di questo articolo è mostrare alcune figure, a prima vista inattese, che si ottengono eseguendo un semplice programma, e spiegare i motivi della loro comparsa. Poche formule matematiche e alcuni facili concetti intuitivi sono sufficienti a giustificare qualitativamente e quantitativamente le caratteristiche di queste strane figure. La conoscenza di questi concetti è utile per chiunque debba rappresentare grafici di funzioni con il calcolatore.

Analisi delle simmetrie

Tutto cominciò quando, facendo girare un mio programma per grafici tridimensionali, ottenni la figura 1. La funzione della quale volevo disegnare il grafico era $z = \sin(r^2)/r^2$ (dove $r^2 = x^2 + y^2$). Questa funzione, poiché r è l'unica variabile, ha una simmetria circolare intorno all'origine delle coordinate x e y, che nel grafico si trova al centro della figura. Simmetria circolare vuol dire che, se si traccia una circonferenza di raggio R qualsiasi con centro nell'origine, su tutti i punti della circonferenza la funzione ha lo stesso valore.
Sul grafico ottenuto dal programma si può osservare molto bene la simmetria circolare nella zona centrale della figura, mentre si vede che ai bordi questa simmetria diventa confusa e non è più riconoscibile. Sembra quasi che le piccole gobbe, che si vedono chiaramente, prendano il sopravvento su di essa, e inoltre che queste gobbe tendano a disporsi circolarmente intorno a un punto (in alto, fuori dallo schermo) diverso dall'origine. La situazione è simile a quella della figura 3b, nella quale si vede come la struttura ad anelli circolari concentrici intorno al centro della figura manifesti delle irregolarità verso i bordi del disegno.

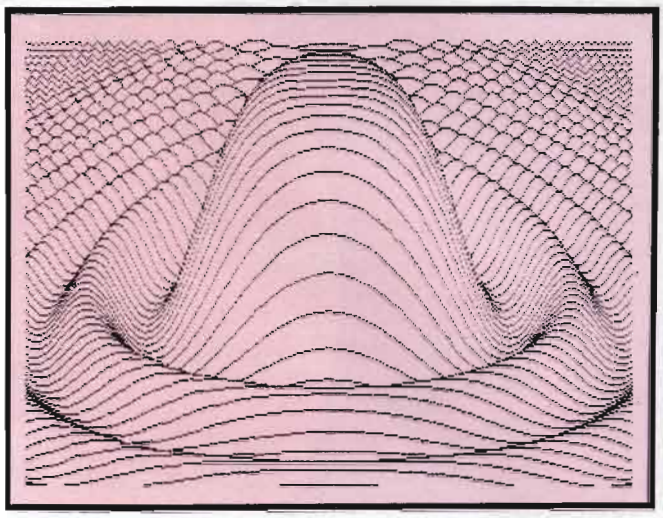


Figura 1. *Grafico tridimensionale della funzione sin $(r^2)/r^2$. Si può notare come le "gobbe" nella parte alta della figura sembrino essere disposte circolarmente intorno ad un punto più in alto del margine del disegno.*

Come mai la simmetria circolare è in parte scomparsa? Dapprima pensai ad una illusione ottica; poi, riflettendo un momento, cominciai a sospettare quale fosse il motivo di questa inattesa rottura di simmetria; questo sarà l'oggetto della discussione seguente.
Per prima cosa, conviene cercare di rappresentare meglio tutti i particolari della situazione. Mi limiterò a considerare la "proiezione" della funzione sin (r^b) sul piano x, y, rappresentandone il segno con un puntino nero se positivo, bianco se negativo; per fare ciò userò il programma del listato 1, il quale, definita una funzione z all'interno della subroutine 1000, consente di rappresentare il segno su un settore del piano xy, delimitato dalle rette $x = XM$, $x = -XM$, $y = YM$, $y = -YM$ (figura 2a), dove i valori di XM e di YM sono ogni volta scelti e dati in input al programma. Com'è ovvio, ogni funzione può essere calcolata solo in un numero finito di punti: questi punti sono rappresentati dalle intersezioni delle linee tratteggiate orizzontali e verticali nel reticolo di figura 2b, che è solo una porzione del reticolo completo 216 x 216. Il valore della funzione verrà dunque calcolato esclusivamente in questi punti; e, come ripeto, ciascuno dei 216 x 216 puntini visualizzati sullo schermo è colorato in nero se il valore della funzione in quel punto è maggiore o uguale a 0, è invece bianco se la funzione è

Indagine su alcune strane figure

negativa.
La distanza tra un punto del reticolo ed il successivo vale $\Delta x = 215/2XM$ nella direzione dell'asse x, e $\Delta y = 215/2YM$ nella direzione dell'asse y. Per comodità, d'ora in poi sarà sempre $XM = YM$, e quindi $\Delta x = \Delta y$.
Per semplificare una varietà di situazioni, riporto qui di seguito le figure 3a-3p, ottenute usando la funzione $z = \sin(r^b)$ e variando sia l'esponente b che il parametro XM.
La figura 3a è un buon esempio di quello che ci si aspetterebbe veder comparire sul video; la proiezione di una funzione oscillante in segno in dipendenza della sola variabile r, e con periodo di oscillazione sempre più corto al crescere di r.
Osservando però le altre figure, soprattutto dalla figura 3 c in avanti, ci si accorge che sul video avvengono cose stranissime, totalmente diverse da quello che ci si aspettava. È come se, invece della funzione $\sin(r^b)$, il programma ne usasse un'altra, completamente diversa.
Dall'osservazione di queste figure si nota come la simmetria circolare, e cioè la struttura ad anelli concentrici attorno al centro delle figure, è perduta mano a mano che ci si allontana dall'origine. Ciò è tanto più evidente quanto più è grande XM (e, in ultima analisi, quanto più grande è Δx, il passo del reticolo di figura 2b).
Al posto della vecchia simmetria intervengono altri tipi di simmetrie e di regolarità, che adesso analizzerò in dettaglio.
Tre cose balzano subito evidenti agli occhi:
1) in ognuna di queste figure, anche se la simmetria circolare è andata perduta, è rimasta una simmetria tale che, invertendo la figura rispetto a uno degli assi o ad una diagonale, la figura non cambia;
2) oltre alla simmetria circolare che si può osservare al centro di ogni figura, vi sono, in alcune di queste figure, altre zone localmente simmetriche distribuite con una certa regolarità. Queste zone hanno dimensioni uguali a quella centrale quando $b = 2$, minori quando $b > 2$, e maggiori quando $b < 2$;
3) in alcune di queste figure, tra le varie zone a simmetria circolare bene evidenti, se ne notano altre meno definite, e si può osservare come anche queste abbiano una distribuzione regolare.

Mi propongo ora di dare una spiegazione a tutte queste apparenti stranezze. Per far questo cercherò di chiarire successivamente i seguenti punti:
1) come mai la simmetria circolare è rotta in modo

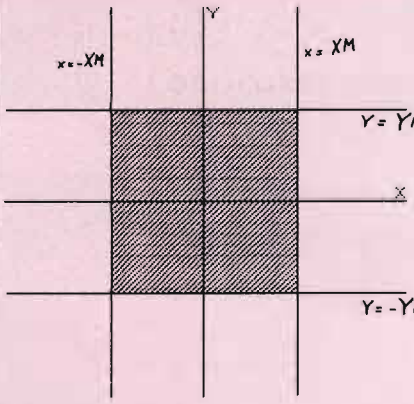


Figura 2a. *La parte tratteggiata del piano x, y rappresenta il campo sul quale viene rappresentato il segno della funzione data.*

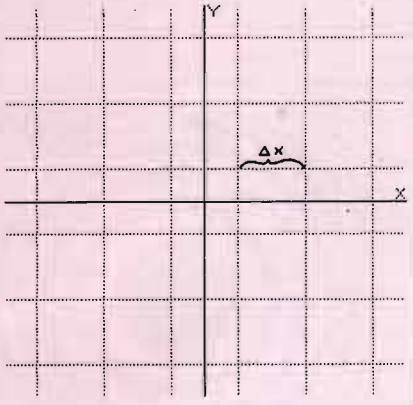


Figura 2b. *I punti di intersezione delle linee orizzontali e verticali del reticolo tratteggiato sono i punti per i quali il programma calcola il valore della funzione. Δx è la variazione in x al passare da un punto del reticolo al successivo, e vale $\Delta x = 2 \star XM/215$.*

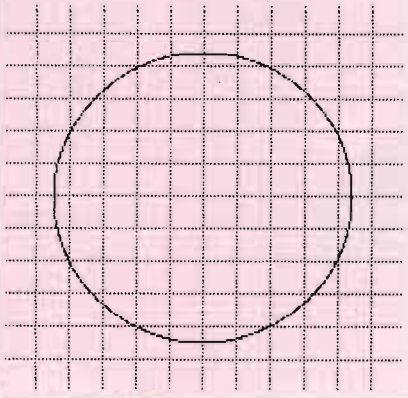


Figura 2c. *I punti del reticolo che corrispondono ad un dato valore di r sono pochi.*

Indagine su alcune strane figure

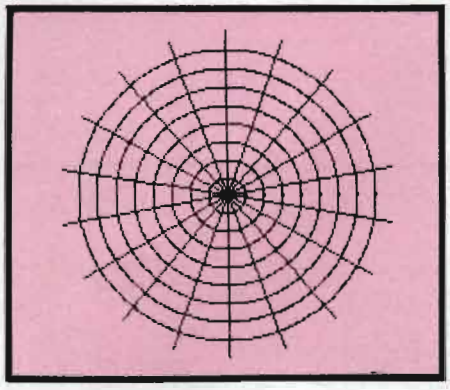
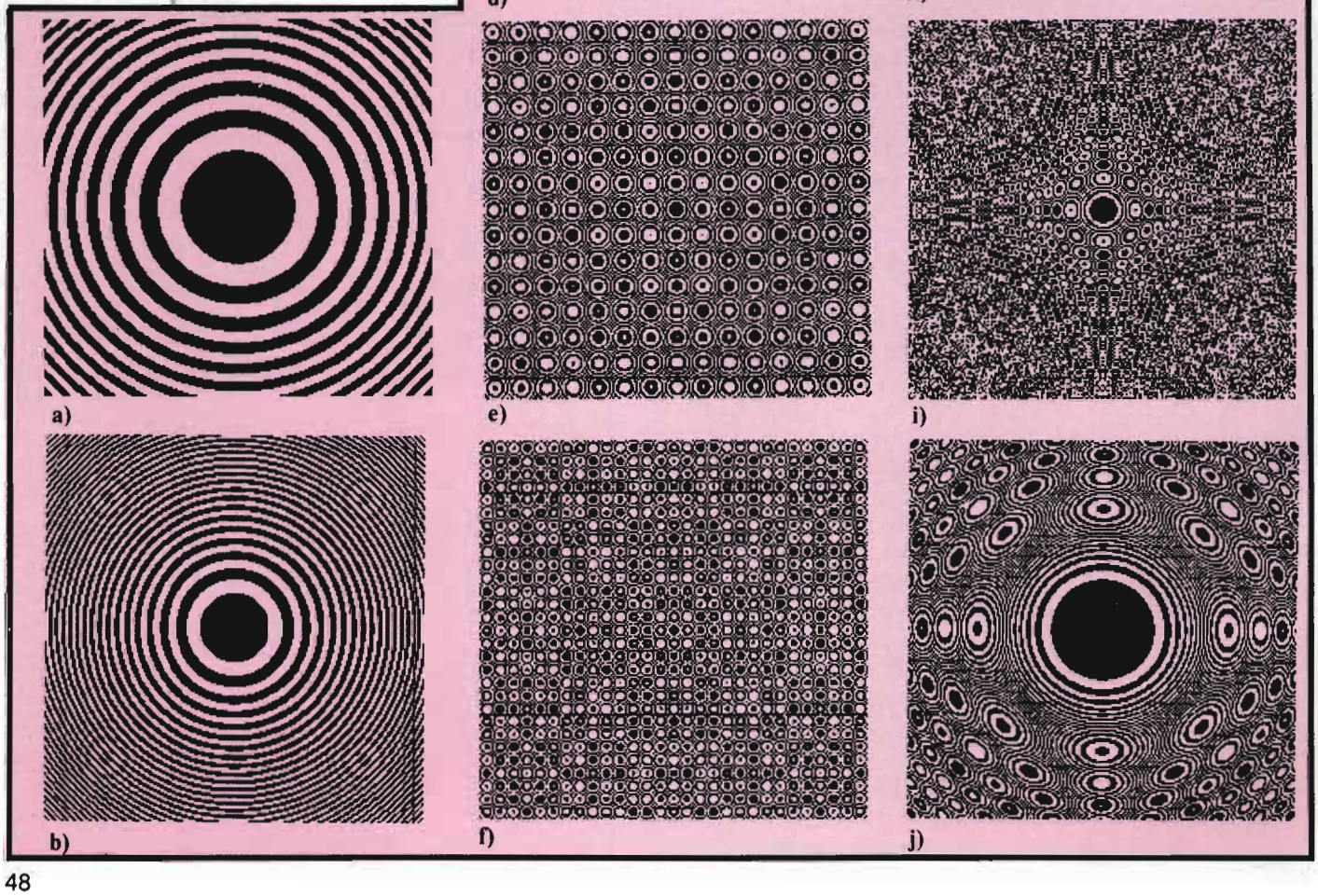
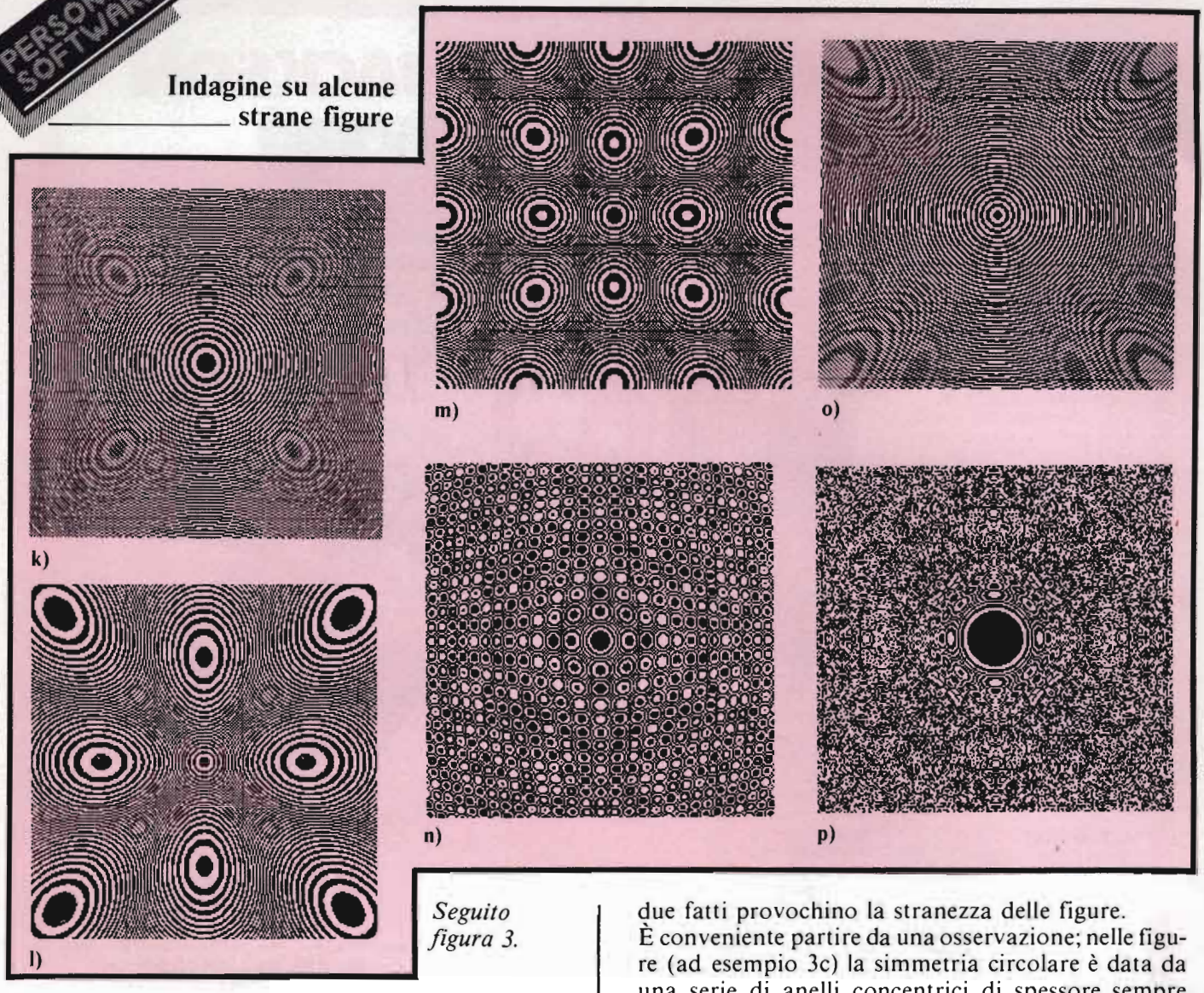


Figura 2d. *Un reticolo di questo tipo sarebbe più adatto per rappresentare una funzione a simmetria circolare.*

Figura 3a-p. *Le figure ottenute facendo girare il programma 1 con diversi valori di b e di XM.*





Seguito figura 3.

così vistoso;
2) come mai rimane una simmetria quadrata;
3) qual'è la ragione per cui si hanno, in parecchi casi, simmetrie circolari centrate in punti diversi dall'origine.

Perché la simmetria circolare sparisce

Il motivo di questa inattesa e impressionante rottura di simmetria è legato a due fatti:
1) il reticolo usato per rappresentare la funzione ha simmetria quadrata e non circolare e inoltre esso è discreto e non continuo (come già detto, la funzione è calcolata solo in 216 x 216 punti);
2) abbiamo a che fare con una funzione (sin r^b) che, quando b è maggiore di 1, al crescere di r oscilla molto rapidamente in segno (cioè, preso un valore di r sufficientemente grande, l'intervallo Δ r, tale che il segno della funzione calcolato in r e quello calcolato in r + Δ r siano opposti, diventa piccolo a piacere).
Nel seguito della discussione chiarirò come questi due fatti provochino la stranezza delle figure.
È conveniente partire da una osservazione; nelle figure (ad esempio 3c) la simmetria circolare è data da una serie di anelli concentrici di spessore sempre minore via via che ci si allontana dall'origine. Ad una certa distanza dall'origine lo spessore di questi anelli è di un solo puntino, mentre oltrepassata questa distanza la struttura ad anelli scompare (o meglio, ne compare un'altra centrata in un altro punto) e la simmetria circolare si rompe.
È ovvio che non è possibile continuare a fare gli anelli sempre più sottili, perché la risoluzione del video (e del nostro reticolo, e questo è il punto essenziale) è discreta e non continua; la domanda che ci si può allora porre è questa: in che modo si dovrebbe manifestare la simmetria circolare oltrepassata questa distanza dall'origine?

Per rispondere a questa domanda, ricordando che le funzioni dalle quali derivano le figure 3 sono del tipo z = sin (t), con t = r^b, analizzeremo in dettaglio cosa succede quando si rappresenta la funzione sin (t) calcolandone il valore in un numero finito di punti. Per fare ciò, si scelgono un certo numero di valori di t, che per comodità terremo equidistanti tra loro, e si calcola il valore della funzione in corrispondenza dei valori di t scelti.
Dalle figure 6 si nota come, fino a che noi "campio-

Indagine su alcune strane figure

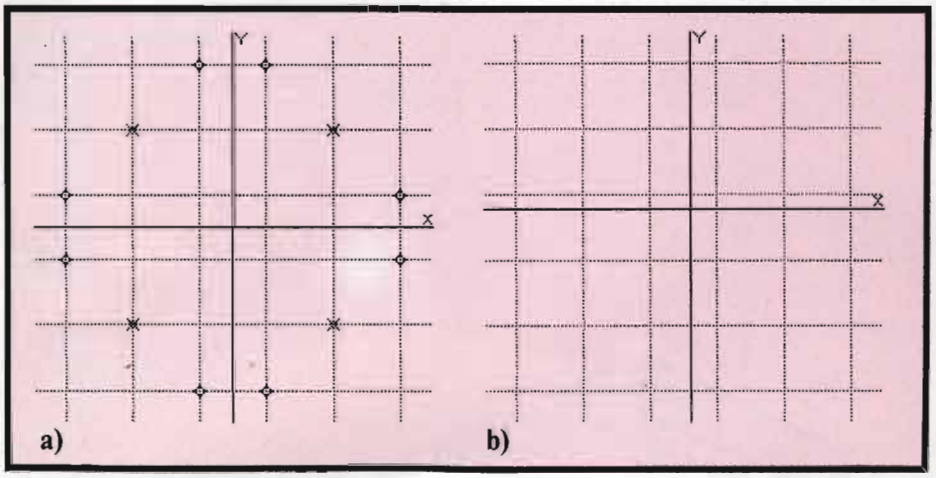


Figura 4. *Il reticolo che definisce i punti per i quali il valore della funzione viene calcolato è centrato intorno all'origine (figura 4a). Ciò fa si che, dato un punto del reticolo, ne esistano anche altri (3 oppure 7) con lo stesso valore di r. Poiché la funzione scelta dipende solo da r, si ha così la simmetria quadrata descritta nel testo.*
Se decentriamo il reticolo, ad esempio spostando l'origine delle coordinate, questa corrispondenza si perde (figura 4b) e si perde anche la simmetria quadrata (figura 5).

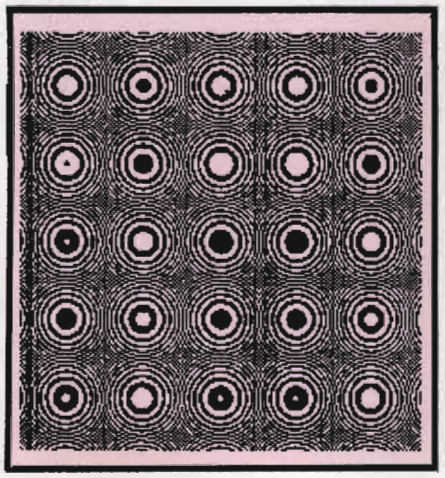


Figura 5. *La stessa figura di 3d, ma questa volta il reticolo non è centrato nell'origine.*
Si può notare come sia andata perduta la simmetria quadrata.

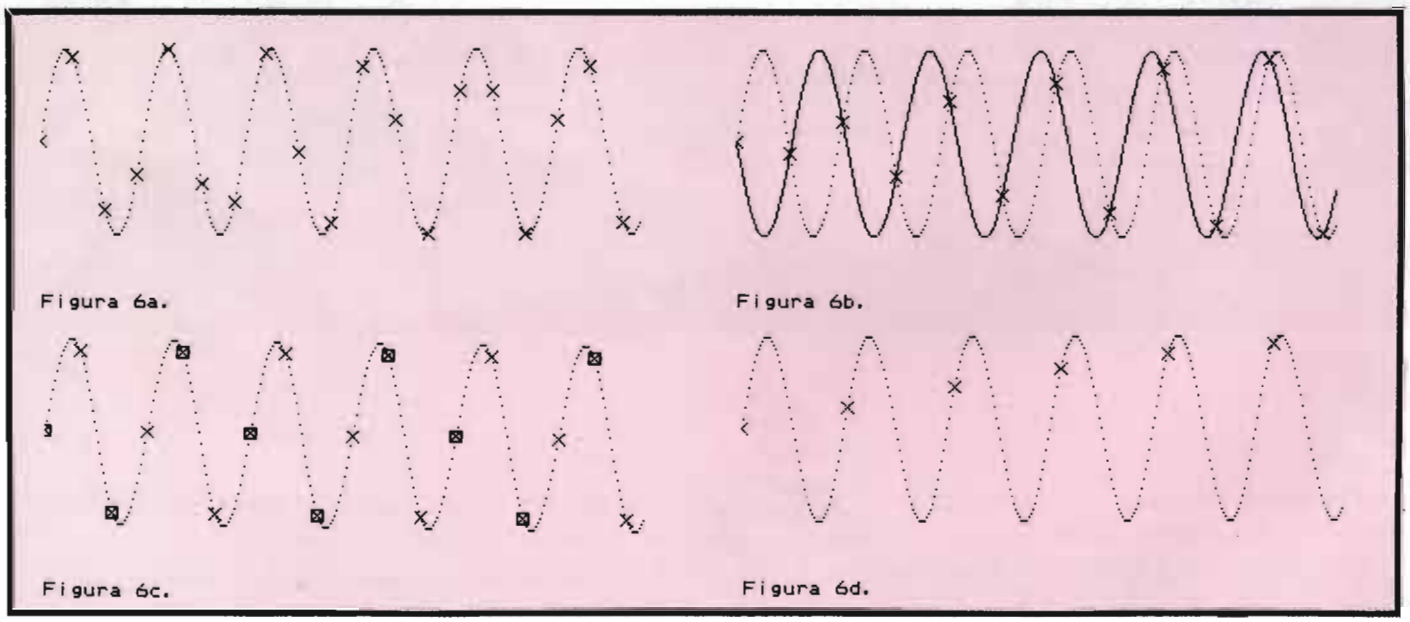


Figura 6. *Fino a che la distanza tra i punti che prendiamo per rappresentare la sinusoide è minore di π (6a), questa può essere ricostruita senza ambiguità a partire da essi. Ma quando la distanza è anche poco maggiore di π, possiamo trovare un'altra sinusoide, con frequenza più bassa della prima, che passa per i punti dati; quindi essi non rappresentano bene la sinusoide originale.*
Si noti come nel caso in cui la distanza è vicina a π (6b), si abbia una alternanza di punti positivi e negativi, nel caso in cui essa vale 120 o 240 gradi (figura 6c., punti rispettivamente marcati con la croce o con il quadrato) si abbiano due punti di un segno seguiti da uno di segno opposto, e infine, quando la distanza vale poco più di 2 π (6d), si abbia una lunga serie di punti dello stesso segno.

Indagine su alcune strane figure

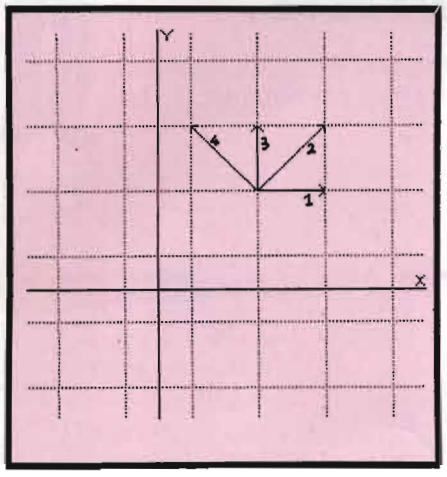


Figura 7. *I diversi modi di passare da un punto del reticolo al punto vicino: essi danno origine alla formula (1.1 - 1.4).*

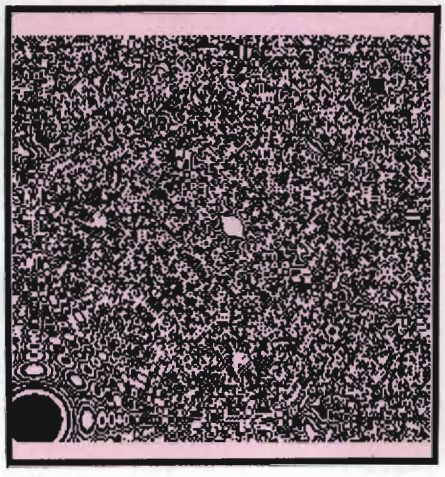


Figura 9. *Per capire come mai compare la zona bianca lungo la diagonale occorrerebbe analizzare anche la derivata dell'intervallo di campionamento al passare da un punto al vicino.*

niamo" questi valori di t in modo che la distanza tra l'uno e l'altro sia minore o uguale alla metà del periodo della funzione, cioè a π, i punti che ricaviamo in questo modo riescono a riprodurre bene tutte le oscillazioni della funzione (la sinusoide a frequenza più bassa che passa per i punti ottenuti è proprio quella che ci è servita per ricavarli) (figura 6a). Ma se la distanza tra un valore di t e il successivo è maggiore di π, i punti ricavati non riescono più a seguire le oscillazioni che la funzione compie (si può trovare una sinusoide con frequenza più bassa della nostra che passa per tutti i punti ottenuti) (figura 6b). In tale caso i punti che abbiamo ricavato riproducono invece molto meglio le oscillazioni di un'altra sinusoide, con periodo più lungo di quella dalla quale li avevamo ricavati.

È interessante vedere che cosa succede in due casi particolari: quando l'intervallo di campionamento è poco minore (o maggiore) di 2π, e quando è uguale a $2\pi/3$.

Nel primo caso (figura 6d) il periodo della sinusoide a frequenza più bassa passante per i punti dati diventa molto grande rispetto a quello della sinusoide originale, e si può osservare come vi sia un gran numero di punti campionati consecutivi con lo stesso segno.

Nel secondo caso (figura 6c) si ha una distribuzione regolare di due punti campionati vicini dello stesso segno, seguiti da uno di segno opposto, e così via.

Abbiamo allora visto come, quando l'intervallo di campionamento è troppo grande, non è possibile riprodurre bene l'andamento di una funzione che oscilla in segno. Come adesso vedremo, questa situazione si manifesta proprio nelle nostre figure.

Nel nostro caso, dato che stiamo rappresentando la funzione $\sin(r^b)$, al posto di t usiamo r^b, e l'intervallo di campionamento, che prima era la distanza tra un valore di t ed il successivo, adesso diventa la differenza dei valori di r^b calcolati in punti adiacenti del reticolo. Come si vedrà nell'analisi seguente, questa differenza non è costante, ma varia al variare di x e di y.

Per semplicità analizziamo per ora solo il caso $b = 2$, e calcoliamo il valore del nostro intervallo di campionamento. Questo è dato dalla variazione di r^2 nel passare da un puntino i, j ai puntini vicini. Ci sono 4 modi diversi di spostarsi da un puntino al vicino (figura 7): esaminiamo ognuno di questi modi

1) $r^2(i+1, j) - r^2(i, j) = r^2(x + \Delta x, y) - r^2(x, y) = x^2 + 2x\Delta x + (\Delta x)^2 + y^2 - x^2 - y^2 = 2x\Delta x + 0((\Delta x)^2)$ (1.1)

dove con $0((\Delta x)^2)$ indico termini del secondo ordine in Δx, perciò trascurabili

2) $r^2(i+1, j+1) - r^2(i, j) = 2x\Delta x + 2y\Delta x$ (1.2)

3) $r^2(i, j+1) - r^2(i, j) = 2y\Delta x$ (1.3)

4) $r^2(i-1, j+1) - r^2(i, j) = 2y\Delta x - 2x\Delta x$ (1.4)

Da questi risultati si vede come l'intervallo di campionamento dipenda dai valori di x e y (oltre che da Δx, che però è costante). Procedendo ad esempio lungo la retta $y = 0$, e allontanandosi dall'origine, x cresce, e con esso crescono anche gli intervalli di campionamento nelle direzioni 1, 2 e 4 fino ad oltrepassare il valore di π; oltre questo punto l'andamento della funzione non è più ben rappresentato dai punti disegnati sul video. Questo del resto lo avevamo già intuito notando come le larghezze degli anelli circolari, che rappresentano gli intervalli di r per i quali la funzione conserva lo stesso segno, diminuiscano a mano a mano che ci si allontana dal centro della figura. Quando gli anelli hanno spessore uguale ad un puntino, siamo nella zona in cui l'intervallo di campionamento nella direzione del raggio è vicino a π (la funzione cambia segno ad ogni puntino). Oltrepassato il raggio per il quale l'intervallo è uguale a π, la struttura ad anelli via via più sottili non può più rappresentare bene la funzione.

Indagine su alcune strane figure

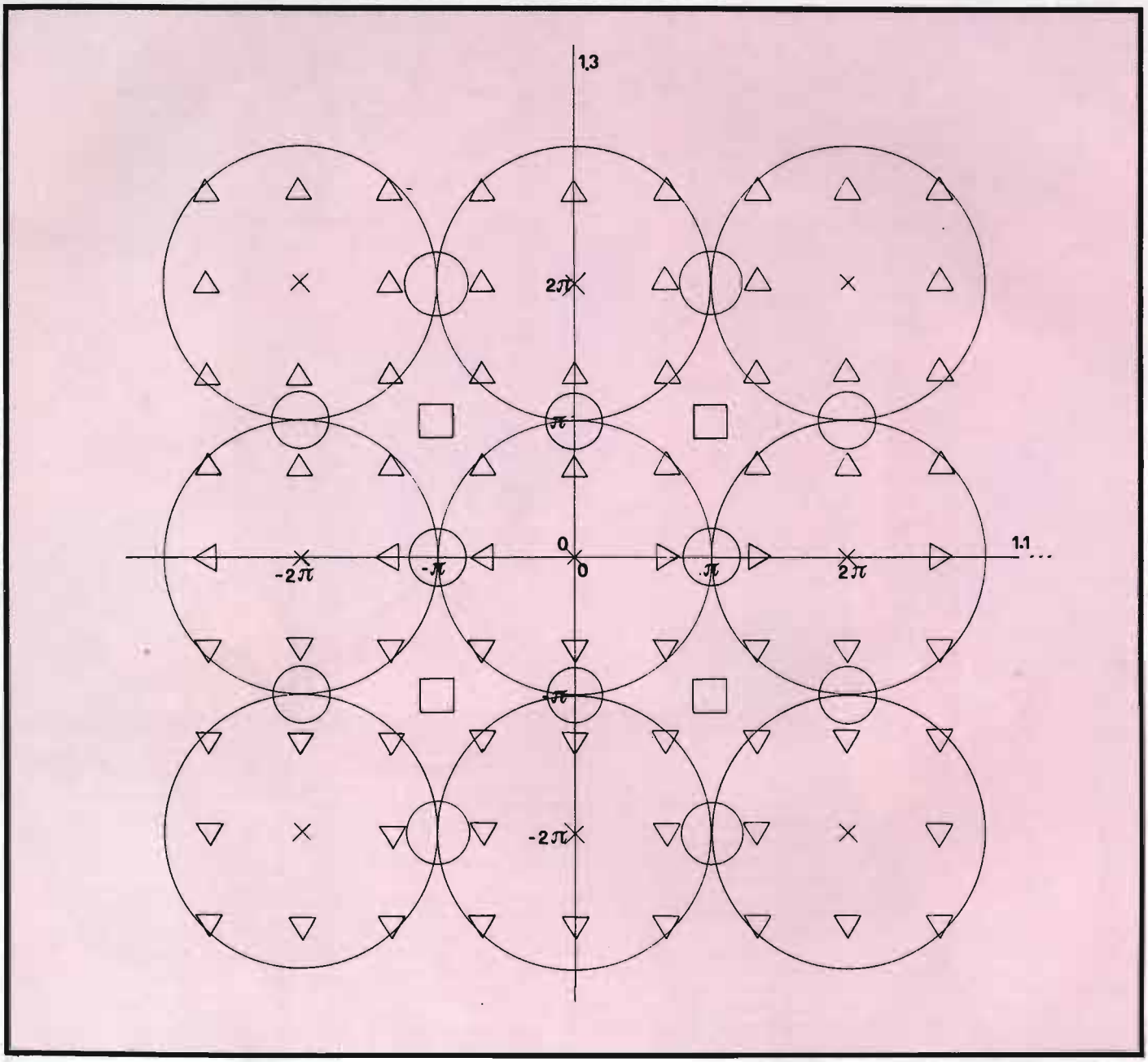


Figura 8. *Lo schema di figura illustra la struttura delle zone di simmetria più evidenti visibili, ad esempio, in figura 3c. I punti marcati con la croce sono quelli nei quali i 4 valori (1.1 - 1.4) degli intervalli di campionamento sono multipli interi di 2 π, e i cerchi che li contengono rappresentano l'estensione delle zone a simmetria circolare centrate in questi punti.*

In tabella 1 sono riportati, per b = 2 e per diversi valori di XM, i valori di x per i quali succede quanto detto; si può facilmente verificare come le figure siano in pieno accordo con questi risultati. Si noti, ad esempio, come nel caso della figura 3a, nella quale la simmetria circolare si conserva indisturbata, il punto in questione giaccia molto fuori dello schermo, mentre nella figura 3b si comincino a notare delle irregolarità dovute alla vicinanza del punto in questione (anche in questo caso fuori dello schermo).

Sorge però adesso una domanda imbarazzante; abbiamo visto cosa succede sull'asse x, e come, oltrepassata una certa distanza dall'origine, la distanza tra i punti del reticolo renda impossibile riprodurre bene l'andamento del segno della funzione. Questo non spiega ancora come mai, dato che la funzione dipende solo dalla distanza dall'origine, il suo comportamento non è uguale in tutte le direzioni, e cioè

perché quello che avviene sull'asse x non si riproduca esattamente su un altro qualsiasi raggio.
La causa di questa rottura di simmetria sta nel fatto che, anche se la funzione che viene rappresentata ha una simmetria circolare, il reticolo che si usa per definire i punti del piano xy nei quali il valore della funzione verrà calcolato per rappresentarla non possiede questa simmetria. In altre parole: è vero che il valore della funzione dipende solo da r, ma nel reticolo che usiamo, pochi punti hanno lo stesso valore di r (figura 2c). Se invece di un reticolo come quello che usiamo (al quale siamo del resto vincolati perché il reticolo dei puntini che possiamo visualizzare sul video ha la stessa forma) potessimo usare il reticolo di figura 2d, potremmo continuare ad osservare una simmetria circolare nella figura.
In pratica, usando un reticolo quadrato invece che quello circolare, si commette un errore dello stesso tipo di quello che si commetterebbe se nelle figure 6, invece di usare per il campionamento i valori di t dai quali abbiamo ricavato i punti marcati con le croci, avessimo usato degli altri valori, ognuno dei quali compreso nell'intervallo tra due dei valori originali, ma non ad una distanza fissa tra essi.

Ma allora, visto che il reticolo usato è inadeguato a rappresentare una funzione a simmetria circolare, come mai nella parte centrale delle figure 3 questa simmetria continua ad essere osservata, e come mai nella figura 3a non si osserva alcuna rottura di simmetria?
La risposta è abbastanza semplice: nella parte centrale della figura la situazione è analoga a quella di figura 6a, e il fatto di usare un puntino del reticolo quadrato invece che uno del reticolo circolare non modifica sensibilmente la rappresentazione della funzione; invece, dove la distanza tra due punti vicini del reticolo è dello stesso ordine di grandezza del periodo della funzione (figura 6d), l'errore che si commette usando il reticolo quadrato può alterare completamente la rappresentazione.

La simmetria residua rispetto agli assi ed alle diagonali

Questa simmetria residua, osservabile in ognuna delle figure 3, deriva dal fatto che, poiché il reticolo di punti sul quale calcoliamo il valore della nostra funzione è centrato nell'origine delle coordinate, per ogni punto del reticolo (x, y) associato ad una coppia di valori (i, j) esistono effettivamente sul reticolo i punti ad esso simmetrici rispetto agli assi e alle diagonali (figura 4a). I valori di x e di y in questi altri punti si possono ottenere o scambiando i segni di x, di y o di entrambi, oppure scambiando x con y (e poi scambiando i segni come nel caso precedente).
Poiché la funzione dipende solo da r, e cioè da $x^2 + y^2$, è chiaro che nei punti suddetti il valore sarà lo stesso. Per eliminare questa simmetria, basta spostare l'origine delle coordinate in modo da distruggere questa corrispondenza (figura 4b). Ciò si ottiene molto facilmente ad esempio modificando la linea:

1023 X = X + 0.05 : Y = Y + 0.09

Facendo allora girare il programma con b = 2 e XM = YM = 30, si ottiene la figura 5; si può notare come la simmetria quadrata sia scomparsa: ad esempio, le zone centrali dei 4 cerchi sopra, sotto e di fianco a quello contenente l'origine sono diverse tra loro, mentre prima erano uguali (figura 3d).

$2 \times \Delta x = \pi$ ($\Delta x = 2XM/215$)

da cui $x = \pi/2\Delta x$

XM	x
6	28.14
10	16.89
20	8.44
30	5.63
50	3.38
70	2.41
100	1.69

Tabella 1. *Distanza dall'origine oltre la quale non è più riconoscibile la simmetria circolare (figure 3a-3g).*

Perché compaiono altre zone a simmetria circolare

Dalle formule (1.1) - (1.4) si nota come esistano dei punti nei quali ognuno dei 4 intervalli di campiona-

Indagine su alcune strane figure

mento vale 0, o 2 π, o un multiplo intero di 2 π; questi punti sono quelli per i quali sia x che y si possono scrivere nella forma

$$\frac{2\,n\,\pi}{2\,\Delta\,x},$$

con n intero.

In questi punti si ha una situazione analoga a quella dell'origine; inoltre, poiché la variazione dell'intervallo di campionamento passando da un puntino al vicino è costante in tutto lo spazio (lo si vede differenziando le 1.1 - 1.4), lo spazio necessario perché l'intervallo di campionamento passi da 0 a π è uguale a quello per passare da 2 π a 3 π, e così le dimensioni degli anelli che si vengono a formare intorno a questi punti "privilegiati" sono uguali in tutta la figura.

Il colore e la dimensione degli anelli centrali possono essere diversi in ognuno di questi punti; ciò dipende dallo sfasamento che la funzione sin (r^2) ha in questi punti rispetto all'origine, sfasamento che è facilmente calcolabile:

$$\varphi = r^2 \bmod 2\,\pi = \left[\frac{4\,\pi^2}{4\,(\Delta\,x)^2}\,(n^2{}_x + n^2{}_y)\right] \bmod 2\,\pi$$

In figura 8 è riportato uno schema che rappresenta la struttura delle nostre figure in funzione degli intervalli di campionamento 1 e 3 (gli intervalli 2 e 4 si ottengono rispettivamente sommando e sottraendo 1 e 3). In questo schema i punti marcati con "x" rappresentano i punti dei quali si è detto sopra, e i cerchi che li contengono sono le zone a simmetria circolare ben definita intorno a questi punti.

Sulle figure notiamo, molto evidenti anche se meno delle prime, altre zone con una certa simmetria circolare. Queste zone si trovano nei punti dove 3 degli intervalli di campionamento valgono +/− 2 π/3, + 2 n π, ed il restante 2 n π. Sul nostro schema contrassegnamo questi punti con un triangolo.

Queste zone sono meno evidenti delle precedenti sia perché quando l'intervallo di campionamento vale 2 π/3 non si ha un gran numero di punti positivi seguiti da un gran numero di punti negativi, ma la distribuzione a due punti di un colore seguiti da un punto del colore opposto già illustrata in figura 6c, sia perché il fatto che uno degli intervalli valga 2 n π fa si che in una direzione il valore della funzione mantenga lo stesso segno per un buon numero di puntini, disturbando la simmetria circolare.

I punti contrassegnati con un cerchietto sul nostro schema sono quelli nei quali 3 degli intervalli valgono 2 n π + π, ed il quarto vale 0 (o 2 n π). Così, mentre l'intervallo uguale a 0 è responsabile delle linee continue che attraversano la figura, i tre intervalli uguali a π fanno si che, nella direzione perpendicolare a queste linee, puntini vicini abbiano segni opposti e, perciò, colori diversi.

Infine, nei punti contrassegnati da un quadratino, gli intervalli 1 e 3 valgono π, e così ogni puntino è di colore opposto a quello immediatamente a fianco, o sopra, o sotto, mentre gli intervalli 2 e 4 valgono 2 n π, dando origine alle linee diagonali formate da puntini dello stesso colore.

Cosa succede quando b è diverso da 2

Osservando le figure 3h-3p, ottenute con valori di b diversi da 2, si possono notare 3 cose:

1) le varie zone che si individuano nelle figure non hanno più una simmetria circolare, ad eccezione della zona centrale;

2) le dimensioni delle zone variano all'interno delle singole figure; in particolare, se b > 2 le zone diventano più piccole allontanandosi dal centro, mentre se b < 2 succede il contrario;

3) anche la densità delle zone varia all'interno delle figure, e si perde la struttura a reticolo ripetitivo che caratterizza il caso b = 2. Quando b > 2 le zone sulle diagonali tendono ad avvicinarsi al centro, mentre quando b < 2 si allontanano da esso.

Tutti e tre questi fatti sono facilmente spiegabili osservando l'andamento degli intervalli di campionamento e delle loro variazioni in funzione di x e y. Generalizzando le espressioni 1.1 - 1.4 si ha:

$$r^n(x+\Delta x, y) - r^n(x,y) = [\,r^2(x+\Delta x, y)\,]_{n/2} - [\,r^2(x,y)\,]_{n/2} =$$

$$= (x^2 + 2\,x\,\Delta\,x + y^2)_{n/2} - (x^2 + y^2)_{n/2} \equiv \Delta\,R_x$$

mentre, per le variazioni dell'intervallo di campionamento:

```
1000  READ B,XM
1002  YM=XM
1010  MODE 6:COLORG 15 3 5 0
1015  FILL 0,0 XMAX,YMAX 15
1020  FOR I=0.0 TO 215.0:FOR J=0.0 TO 215.0
1022  X=I*2.0*XM/215.0-XM:Y=J*2.0*YM/215.0-YM
1024  R=SQR(X*X+Y*Y)
1025  Z=SIN(R^B)
1030  COLOR%=0:IF Z<0 THEN COLOR%=15
1040  DOT I,J COLOR%
1050  NEXT J:NEXT I
1060  END
```

Listato 1. *Programma per il tracciamento dei grafici scritto in BASIC per il DAI, ma facilmente traducibile per altri calcolatori.*

Libri firmati JACKSON

nuovidea

Alan Miller
PROGRAMMI SCIENTIFICI IN PASCAL
Un'opera base per chi desidera costruirsi una "libreria" di programmi in grado di risolvere i più frequenti problemi scientifici e ingegneristici.
372 pagine **L. 25.000**
Codice **554P**

Franco Filippazzi
Giulio Occhini
VOI E L'INFORMATICA
L'opera che il manager moderno non può ignorare. In 100 tavole: gli strumenti dell'Informatica, l'Informatica e l'Azienda, realtà e prospettive tecnologiche...
116 pagine **L. 15.000**
Codice **526A**

Roland Dubois
CAPIRE I MICROPROCESSORI
Un fantastico viaggio alla scoperta del "cervello" elettronico: la funzione del microprocessore, delle memorie ROM e RAM, delle interfacce...
126 pagine **L. 10.000**
Codice **342A**

Giuseppe Saccardi
TRASMISSIONE DATI Dispositivi standard e protocolli
Il calcolatore e le sue infinite applicazioni nel campo delle comunicazioni applicate a tutti i settori in cui si articola la società moderna. Un libro che traduce in tecnologia la profezia orwelliana di "1984"
308 pagine **L. 23.000**
Codice **528P**

F. Franceschini
F. Paterlini
Voi e il vostro Commodore 64
Uno strumento fondamentale per la comprensione e programmazione del Commodore 64.
Con consigli, programmi testati, glossario e utili accenni di BASIC.
256 pagine **L. 22.000**
Codice **347 B**

La Biblioteca che fa testo

In busta chiusa, e senza impegno, inviate questo coupon a: Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

☐ Desidero ricevere gratuitamente il Catalogo Generale della Biblioteca Jackson e informazioni sulle 10 Riviste specialistiche da voi pubblicate.
(allego L. 1.000 in francobolli per contributo spese di spedizione)

☐ Desidero ricevere contrassegno il/i volume/i

_____

_____

(pagherò al ricevimento L. .....
più L. 2.000 per contributo spese di spedizione)

Nome _____ Cognome _____

Via _____

CAP _____ Città _____

Indagine su alcune strane figure

$$\frac{j}{jx}\,\Delta R_{\cdot} = (2x + 2\Delta x)(x^2 + 2x\Delta x + y^2)^{n/2\,-1} - 2x(x^2 + y^2)^{n/2\,-1}$$

$$\frac{j}{jy}\,\Delta R_{\cdot} = (2y)(x^2 + 2x\Delta x + y^2)^{n/2\,-1} - 2y(x^2 + y^2)^{n/2\,-1}$$

Come si vede, la variazione dell'intervallo di campionamento al passare da un punto al vicino dipende da x e da y; se $b > 2$ essa cresce al crescere della distanza dall'origine, e ciò spiega la diminuzione delle dimensioni delle zone nelle figure 3h-3j, 3n e 3p; al contrario, se $b < 2$, la variazione decresce con l'allontanarsi dall'origine, e così le zone diventano più grandi.

La perdita della simmetria circolare delle singole zone dipende invece dal fatto che la variazione dell'intervallo di campionamento calcolata in un dato punto dipende dalla direzione lungo la quale questa variazione è calcolata.

Infine, la deformazione del reticolo è causata dal diverso andamento dell'intervallo di campionamento; in particolare, mentre nel caso $b = 2$ valeva l'uguaglianza:

$$x^b + y^b = r^b,$$

in generale si ha che il primo termine è maggiore del secondo se $b < 2$, mentre è minore se $b > 2$. Questo spiega l'avvicinamento al centro delle zone lungo le diagonali, rispetto alle zone lungo gli assi x e y, nel caso di $b > 2$, ed il loro allontanamento nel caso contrario.

Conclusioni

Spesso problemi apparentemente semplici e privi di difficoltà, come ad esempio tracciare il grafico di una funzione, nascondono complicazioni alle quali occorre dedicare un po' di attenzione. L'analisi svolta in questo articolo è stata volutamente poco rigorosa e molto intuitiva, perché le complicazioni matematiche non facessero perdere di vista il punto centrale del problema, ossia la spiegazione dell'origine delle figure 3a-p, ormai non più tanto strane. In particolare, si sono trascurati gli effetti degli errori di arrotondamento introdotti dal calcolatore nel rappresentare i numeri reali (questi errori non disturbano nel caso $b = 2$, ma non si potrebbero trascurare al crescere di b e di XM), e non si è cercato di capire cosa succede quando anche le variazioni degli intervalli di campionamento subiscono notevoli cambiamenti da un puntino al successivo (figura 9). A volte, come nel nostro caso, alcuni di questi effetti hanno una importanza secondaria e si possono trascurare; altre volte ciò non è possibile, e per risolvere un problema non si può evitare di analizzarne ogni minimo particolare. ■

PERSONAL SOFTWARE

Sprite Editor per il TI99/4 A

Un programma veramente efficiente per definire i caratteri grafici

di *Filippo Cerulo*

Tutti i possessori di TI99/4 A si trovano, prima o poi, di fronte alla necessità di definire nuovi caratteri da utilizzare nei programmi. Il procedimento è relativamente semplice e senz'altro più accessibile al principiante di analoghe costruzioni possibili su altri home computer. Ciò non toglie che quando il numero di caratteri da ridefinire è elevato il calcolo "manuale" dell'identificatore di sagoma esadecimale sia molto noioso e stancante. Inoltre, in programmi che usano sprite a grandezza doppia ogni sagoma relativa è formata da quattro caratteri: quindi deve essere costruita su una griglia formata da 16x16 pixel ed identificata da una stringa esadecimale lunga ben 64 caratteri. A questo punto sembra inevitabile demandare questi gravosi (e tediosi) compiti al proprio computer: questo è esattamente ciò che ho fatto.

Qualche cenno sulla CALL CHAR

L'argomento ridefinizione di caratteri è stato già ampiamente trattato anche sulle pagine di questa rivista ed inoltre il manuale, almeno su questo punto, è abbastanza chiaro.
Nel TI99 tutti i caratteri il cui codice ASCII è compreso tra 32 e 159 possono essere ridefiniti, cioè può essere modificata la loro sagoma. Ad ogni sagoma è associato un *identificatore* formato da sedici cifre esadecimali. La sagoma di ogni carattere viene costruita su una griglia di 8x8 pixel. Ogni cifra esadecimale dell'identificatore rappresenta, secondo la tabella riportata nel manuale, 4 pixel in sequenza da destra a sinistra e dall'alto in basso. Con l'istruzione:
CALL CHAR (codice di carattere, "identificatore di sagoma")
si ridefinisce un carattere. Questa istruzione può essere usata anche come *comando* per i caratteri il cui codice sia compreso tra 128 e 159. Cioè, se si dà ad esempio il comando:
CALL CHAR (129, "3C4299A-1A199423C")
il simbolo di copyright diventa disponibile sulla tastiera premendo CTRL A. In ogni caso, i caratteri il cui codice sia compreso tra 32 e 127 possono essere ridefiniti da programma, ma quando si torna in "modo comando" (cioè ad ogni stop del programma) riprendono le sagome originali.

Le differenze del BASIC esteso

In Extended BASIC possono essere ridefiniti solo i caratteri il cui codice ASCII sia compreso tra 32 e 143. Inoltre l'istruzione CALL CHAR, oltre il modo usuale, ha altre due costruzioni sintattiche. Le illustreremo con due esempi. L'istruzione:
100 CALL CHAR
(65, "080C0EFFFF0E0C083C4299
A1A199423C")
ridefinisce i caratteri "A" e "B" come, rispettivamente, una freccia ed il simbolo di copyright. In BASIC esteso, infatti, l'identificatore può essere lungo fino a 64 cifre esadecimali e ridefinisce il carattere il cui codice è specificato nella istruzione, nonchè i caratteri seguenti, usando sedici cifre esadecimali per ogni carattere. L'istruzione:
CALL CHAR (65, "080C0EFFF-F0E080C", 78, "000000FFFF")
ridefinisce i caratteri "A" ed "N". Possiamo cioè ripetere codici ed identificatori all'interno delle parentesi senza riscrivere la CALL CHAR. Notate, nel caso del codice 78, che gli zeri finali di un identificatore esadecimale possono essere omessi. Un'altra caratteristica interessante (comune anche ad altri BASIC) è la funzione RPT$. Si tratta di una funzione di stringa con la seguente sintassi:
RPT$ ("stringa", numero di ripetizioni)
Ad esempio l'istruzione:
200 A$ = RPT$ ("0C",8)
equivale a:
200 A$ = "0C0C0C0C0C0C0C0C"
In pratica, essa assegna ad una variabile di stringa una ripetizione di caratteri determinati. Può essere usata (come tutte le funzioni di stringa) direttamente in una CALL CHAR, come nell'esempio seguente:
200 CALL CHAR (129, RPT$ ("F", 16))
Questa istruzione ridefinisce il carattere 129, "accendendo" tutti i pixel che lo compongono. Un cenno va fatto anche ad un'altra istruzione molto utile. La sua sintassi è:
CALL CHARPAT (codice di carattere, variabile di stringa)
Essa assegna a "variabile di stringa" l'identificatore esadecimale della sagoma rappresentante il carattere il cui codice ASCII compare nell'istruzione. Ad esempio:
200 CALL CHARPAT (65,A$)
assegna ad A$ il valore esadecimale

Listato 1. *Il programma BASIC.*

```
100 !**** SPRITE EDITOR ****
110 !
120 !(C) 1984 - FILIPPO CERULO - ARCADE
130 !
140 !TEL. 0824/871190
150 !
160 DIM BN$(15),HE$(15)
170 FOR I=0 TO 15 :: READ BN$(I),HE$(I):: NEXT I
180 DATA 0000,0,0001,1,0010,2,0011,3,0100,4,0101,5,0110,6,0111,7
190 DATA 1000,8,1001,9,1010,A,1011,B,1100,C,1101,D,1110,E,1111,F
200 FOR I=104 TO 109 :: READ A$ :: CALL CHAR(I,A$):: NEXT I :: CALL COLOR(10,6,1
)
210 DATA 000000FFFF,1818181818181818,000000F8F8181818,181818F8F8,0000001F1F18181
8,1818181F1F
220 CALL CLEAR :: CALL CHAR(91,"3C4299A1A199423C",93,"000000FF",132,"00003C3C3C3
C",139,"FF818181818181FF",140,RPT$("F",16))
230 CALL COLOR(14,3,1):: CALL SCREEN(15)
240 GOSUB 770 :: CALL CHARPAT(65,B$,66,C$):: CALL SCRIVI(7,3,B$,BN$()):: CALL SC
RIVI(15,3,C$,BN$()):: CALL STAMPA(8):: CALL GET(COD)
250 CALL CLEAR :: GOSUB 770 :: CALL STAMPA(2):: CALL MASCHERA
260 DATA 3,8,"SPRITE  EDITOR",4,8,]]]]]]]]]]]]]]],8,14,EXTENDED BASIC,10,14,SOFTW
ARE,13,14,[ 1984 BY
270 DATA 15,14,ARCADE,19,14,PRESS ANY,21,14,KEY,1,7," SPRITE  EDITOR hhhhhh",2,8
,]]]]]]]]]]]]]]
280 CALL CANCELLA :: RESTORE 290 :: CALL STAMPA(6)
290 DATA 6,19,SCEGLI :,9,19,1 WRITE,11,19,"2 DISPLAY",13,19,3 STRINGA,15,19,"4 N
EW",17,19,5 FINE
300 CALL GET(COD):: IF COD<49 OR COD>53 THEN 300
310 ON COD-48 GOSUB 350,490,560,660,330
320 GOTO 280
330 CALL CLEAR :: STOP ! FINE LOGICA DEL PROGRAMMA
340 REM **** SBR WRITE ****
350 CALL CANCELLA :: RESTORE 360 :: CALL STAMPA(5)
360 DATA 6,19,WRITE,9,20,1 OVER,11,20,2 SET,13,20,3 RESET,15,20,4 STOP
370 CALL MAGNIFY(1):: CALL SPRITE(#2,132,5,33,17):: X=33 :: Y=17 :: CAR=0
380 CALL LOCATE(#2,X,Y)
390 CALL KEY(5,COD,ST):: IF ST=0 THEN 390 ELSE IF POS("1234ESDXWRZC",CHR$(COD),1
)=0 THEN 390
400 IF COD=52 THEN CALL DELSPRITE(#2):: RETURN
410 IF COD>52 THEN 450
420 IF COD=49 THEN CAR=0 :: CALL COLOR(#2,5):: GOTO 390
430 IF COD=50 THEN CAR=140 :: CALL COLOR(#2,2):: GOTO 390
440 IF COD=51 THEN CAR=139 :: CALL COLOR(#2,16):: GOTO 390
450 CALL POSITION(#2,X,Y):: IF CAR<>0 THEN CALL HCHAR(INT(X/8)+1,INT(Y/8)+1,CAR)
460 C$=CHR$(COD):: X=X+8*(-(POS("WER",C$,1)<>0)*(X>40)+(POS("ZXC",C$,1)<>0)*(X<1
46))
470 Y=Y+8*(-(POS("SWZ",C$,1)<>0)*(Y>24)+(POS("DRC",C$,1)<>0)*(Y<130)):: GOTO 380
480 REM **** SBR DISPLAY ****
490 CALL CANCELLA :: DISPLAY AT(3,20):"DISPLAY" :: DISPLAY AT(23,1)SIZE(18)BEEP:
"ATTENDERE PREGO !"
500 RESTORE 510 :: PAT$=""
510 DATA 5,3,5,19,13,3,9,19,5,11,13,19,13,11,17,19
520 FOR I=1 TO 4 :: READ R,C,R1,C1 :: CALL LEGGI(R,C,ID$,HE$()):: PAT$=PAT$&ID$
530 DISPLAY AT(R1,C1):SEG$(ID$,1,8):: DISPLAY AT(R1+2,C1)BEEP:SEG$(ID$,9,8):: NE
XT I
```

Seguito listato 1.

```
540 GOSUB 680 :: RETURN
550 REM **** SBR STRINGA ****
560 CALL CANCELLA :: CALL MASCHERA :: CALL CHAR(128,RPT$("0",64)):: DISPLAY AT(3
,22):"STRINGA"
570 COD=128 :: PAT$="" :: RESTORE 580
580 DATA 5,3,13,3,5,11,13,11
590 FOR RI=5 TO 19 STEP 4 :: CALL HCHAR(23,2,32,18):: CALL CHARPAT(COD,B$):: DI
PLAY AT(RI,19):SEG$(B$,1,8)
600 DISPLAY AT(RI+2,19):SEG$(B$,9,8):: ACCEPT AT(RI,19)SIZE(-8)VALIDATE(DIGIT,"A
BCDEF")BEEP:B1$
610 ACCEPT AT(RI+2,19)SIZE(-8)VALIDATE(DIGIT,"ABCDEF")BEEP:B2$ :: B$=B1$&B2$ ::
PAT$=PAT$&B$
620 READ R,C :: IF B$=RPT$("0",16)THEN 640
630 DISPLAY AT(23,1)BEEP:"ATTENDERE PREGO !" :: CALL SCRIVI(R,C,B$,BN$())
640 COD=COD+1 :: NEXT RI :: GOSUB 680 :: RETURN
650 REM **** SBR NEW ****
660 CALL CANCELLA :: DISPLAY AT(23,4)BEEP:"NEW" :: CALL MASCHERA :: CALL HCHAR(2
3,4,32,15):: RETURN
670 REM *** MOSTRA GLI SPRITES ***
680 CALL CHAR(128,PAT$):: CALL HCHAR(23,3,32,28)
690 DISPLAY AT(23,1)SIZE(8):"COLORE ?" :: ACCEPT AT(23,11)SIZE(2)VALIDATE(DIGIT)
BEEP:COL
700 IF COL>16 THEN 690 ELSE IF COL=0 THEN CALL DELSPRITE(#1):: RETURN
710 CALL HCHAR(22,20,32,10):: CALL HCHAR(23,20,32,10):: CALL DELSPRITE(#1)
720 CALL COLOR(13,COL,1):: DISPLAY AT(22,25):"";"" :: DISPLAY AT(23,25):"";"
"
730 CALL SPRITE(#1,128,COL,1,161,10,0):: CALL MAGNIFY(4)
740 CALL COINC(#1,161,161,1,A):: IF A=0 THEN 740
750 CALL MOTION(#1,0,0):: CALL LOCATE(#1,161,161):: GOTO 690
760 REM **** STAMPA CORNICE ****
770 CALL HCHAR(24,1,104,64):: CALL VCHAR(1,32,105,48):: CALL HCHAR(1,1,108):: CA
LL HCHAR(1,32,106)
780 CALL HCHAR(24,1,109):: CALL HCHAR(24,32,107):: RETURN
790 REM **** SUB STAMPA ****
800 SUB STAMPA(NU):: FOR I=1 TO NU :: READ R,C,S1$ :: DISPLAY AT(R,C):S1$ :: NEX
T I :: SUBEND
810 REM **** SUB GET ****
820 SUB GET(COD):: CALL SOUND(100,1047,6)
830 CALL KEY(5,COD,ST):: IF ST=0 THEN 830
840 SUBEND
850 REM **** SUB MASCHERA ****
860 SUB MASCHERA :: FOR R=5 TO 20 :: CALL HCHAR(R,3,139,16):: NEXT R :: SUBEND
870 REM **** SUB CANCELLA ****
880 SUB CANCELLA :: FOR R=3 TO 23 :: CALL HCHAR(R,20,32,12):: NEXT R :: CALL HCH
AR(23,2,32,30):: CALL HCHAR(22,2,32,30):: SUBEND
890 REM **** SUB BIN ****
900 SUB BIN(A$,N):: N=0 :: FOR I=3 TO 0 STEP -1 :: N=N+VAL(SEG$(A$,4-I,1))*2^I :
: NEXT I :: SUBEND
910 REM **** SUB LEGGI ****
920 SUB LEGGI(R,C,A1$,B1$()):: A1$="" :: FOR R1=R TO R+7 :: FOR C1=C TO C+7 STEP
 4 :: A$="" :: FOR C2=C1 TO C1+3
930 CALL GCHAR(R1,C2,COD):: A$=A$&CHR$(COD-91):: NEXT C2
940 CALL BIN(A$,N):: A1$=A1$&B1$(N):: NEXT C1 :: NEXT R1 :: SUBEND
950 REM **** SUB ESA ****
960 SUB ESA(R,C,A$):: FOR I=1 TO 4 :: CALL HCHAR(R,C-1+I,ASC(SEG$(A$,I,1))+91)::
 NEXT I :: SUBEND
970 REM **** SUB SCRIVI ****
980 SUB SCRIVI(R,C,A$,A1$()):: A=1 :: FOR R1=R TO R+7 :: FOR C1=C TO C+7 STEP 4
:: B$=SEG$(A$,A,1)
990 IF ASC(B$)<58 THEN K=VAL(B$)ELSE K=ASC(B$)-55
1000 CALL ESA(R1,C1,A1$(K)):: A=A+1 :: NEXT C1 :: NEXT R1 :: SUBEND
```

"003844447C444444". Infine l'istruzione:

200 CALL CHARSET

usata in un programma ricarica le sagome normali in caratteri il cui codice sia compreso tra 32 e 127 e che siano stati precedentemente modificati.

Costruire uno sprite

Il programma che vi presento ha come scopo principale quello di calcolare l'identificatore esadecimale relativo alle sagome (esattamente quattro) che potete costruire, attraverso facili procedure, su una griglia di 16x16 pixel ingrandita otto volte. Alla fine il computer vi mostra lo sprite (o il carattere) creato in movimento e con il colore voluto, sia in grandezza normale che doppia. Potete inoltre far costruire sulla griglia il carattere relativo ad un identificatore da voi inserito (anche eventualmente per modificarlo). Vediamo un po' come operare.

Dopo la sigla il computer stampa la griglia ed il menu principale. Le scelte sono accettate tramite istruzione CALL KEY e non INPUT, per evitarvi di premere ENTER ogni volta: quindi non vi aspettate un cursore lampeggiante, ma premete semplicemente il tasto numerico relativo. Esaminiamo ora, una per una, le opzioni possibili.

1 WRITE: vi permette di costruire sulla griglia la sagoma desiderata. La "guida" è uno sprite, a forma di quadratino, che potete muovere utilizzando in modo usuale i tasti W, E, R, S, D, Z, X, C. La guida è sufficientemente veloce e può muoversi anche in diagonale. In modo WRITE (scrivi) sono possibili quattro opzioni e precisamente: OVER (1): il quadratino è blu e si muove sulla griglia senza altre conseguenze; SET (2): il quadratino è nero e "accende" (setta) i pixel al suo passaggio; RESET (3): il quadratino è bianco e "spegne" al suo passaggio i pixel eventualmente accesi; FINE (4): ritorna al menu principale.

Per selezionare l'opzione desiderata basta premere il tasto numerico relativo. Con un po' di pratica la fase di "costruzione" della sagoma diventa abbastanza semplice e perfino divertente. Potete creare una sagoma formata da quattro caratteri (per uno sprite "double sized") o anche quattro sagome diverse, ognuna di un carattere.

2 DISPLAY: calcola l'identificatore esadecimale della sagoma presente sulla griglia. Le cifre esadecimali sono mostrate a gruppi di otto, sulla destra dello schermo, e a ogni due gruppi corrisponde un carattere secondo lo schema:

1 3
2 4

cioè dall'alto in basso e da destra a sinistra. Quindi il computer vi chiede il colore con il quale visualizzare la sagoma. Dovrete perciò introdurre il codice del colore desiderato (un intero tra uno e sedici) ed il computer vi mostrerà in basso a destra la sagoma in grandezza normale, mentre uno sprite in grandezza doppia si muoverà in verticale e si fermerà alla base dello schermo. A questo punto potrete cambiare ancora il colore della sagoma o tornare al menu principale, introducendo zero come codice di colore. In ogni caso la sagoma sulla griglia non sarà modificata.

3 STRINGA: questa opzione vi permette di visualizzare sulla griglia la sagoma relativa ad un identificatore da voi inserito. Ogni eventuale sagoma preesistente viene cancellata e dovete introdurre il vostro identificatore sulla destra dello schermo a gruppi di otto cifre. Ogni sedici cifre viene visualizzato un carattere secondo la solita corrispondenza. Alla fine potrete vedere la vostra sagoma in grandezza normale o doppia, secondo le modalità dell'opzione DISPLAY.

4 NEW: cancella la sagoma eventualmente esistente sulla griglia.

5 FINE: pone termine al programma.

Dopo l'esecuzione di uno qualsiasi dei sottoprogrammi DISPLAY e STRINGA potete tornare in modo WRITE per correggere la sagoma e ricalcolarne l'identificatore. Infatti, in generale, la prima volta che si costruisce una sagoma, essa non sarà esente da imperfezioni. Inoltre, la scelta del colore è fondamentale nell'esaltare o meno alcuni particolari. Considerando anche che il colore di sfondo durante il programma è grigio, vi consiglio di evitare, quando valutate la sagoma, il nero e di orientarvi su colori leggermente più sfumati (blu o verde scuro).

Quattro passi nel listato

Le caratteristiche peculiari dell'Extended BASIC, rispetto ai linguaggi implementati su altri home computer, sono molte. Oltre alla grafica ed agli sprite, infatti, disponiamo di una grande versatilità nelle istruzioni di INPUT/OUTPUT su video e soprattutto abbiamo la possibilità di scrivere sottoprogrammi in BASIC indipendenti dal programma principale e sostanzialmente diversi dalle normali subroutine. Questi sottoprogrammi sono attivati da istruzioni del tipo:

200 CALL nome di sottoprogramma (lista di parametri)

Lo spazio non mi permette di dilungarmi oltre su questi aspetti, peraltro interessantissimi, del BASIC esteso e rimando gli approfondimenti ad altre occasioni. Basti sapere che le variabili usate nei sottoprogrammi sono "esclusive", nel senso che non hanno nulla a che vedere con eventuali altre variabili con lo stesso nome presenti nel programma principale.

Cominciamo appunto ad esaminare i sottoprogrammi usati. Vi accorgerete che alcuni sono di utilizzo generale e possono essere trasportati in altri programmi, esattamente come sono scritti.

SUB STAMPA(NU) - Serve a stampare una serie di stringhe (esatta-

mente NU), inserite nel programma con istruzioni DATA, con la sintassi:
DATA riga, colonna, stringa (,....)
Risparmia molto spazio e permette una facile impostazione di maschere video o di menu anche molto complessi.
SUB GET (COD) - Simula l'istruzione GET (o INKEY$) presente su altri computer. Il programma si ferma, emette un suono e aspetta che premiate un qualsiasi tasto. Il codice ASCII del tasto premuto è ritornato nella variabile COD.
SUB MASCHERA - Stampa la griglia su cui costruire la sagoma.
SUB CANCELLA - Serve a cancellare la parte destra ed il bordo inferiore dello schermo.
SUB BIN(A$,N) - Converte un numero binario di quattro cifre, contenuto nella variabile di stringa A$, nel corrispondente decimale N. È un sottoprogramma di uso universale.
SUB LEGGI (R,C,A1$, B1$()) - Calcola l'identificatore di sagoma del carattere presente su una parte (8x8 pixel) della griglia. Accetta in input le coordinate (R,C), da cui cominciare il calcolo, ed il vettore B1$(I), che corrisponde a HE$(I) del programma principale. Ritorna in A$ l'identificatore. Legge i caratteri sul video con una serie di CALL GCHAR e costruisce ogni volta un numero binario di quattro cifre. Quindi converte il numero binario nel decimale N: il valore esadecimale corrispondente è memorizzato nella variabile B1$(N).
SUB ESA (R,C,A$) - Visualizza sulla griglia i quattro pixel relativi al numero binario contenuto nella variabile A$ indicando la riga e la colonna da cui cominciare.
SUB SCRIVI (R,C,A$,A1$()) - Visualizza sulla griglia il carattere corrispondente all'identificatore contenuto nella variabile A$ indicando la riga e la colonna da cui cominciare. Il vettore A1$(I) corrisponde a BN$(I) del programma principale.

Diamo ora un'occhiata al listato

Le righe 170 - 190 caricano i contenuti dei vettori BN$(I) e HE$(I). Le righe 200 - 230 ridefiniscono i caratteri necessari e determinano i colori dei caratteri e dello sfondo.
La sigla è compresa nella riga 240: i caratteri che in essa appaiono ingranditi sono "A" e "B" (codici 65 e 66), ma ognuno di voi può personalizzare il programma, modificando i codici ASCII nell'istruzione CALL CHARPAT.
La riga 250 stampa la griglia e l'intestazione e la 280 il menu principale.

Dopo la scelta, accettata alla linea 300, il programma passa il controllo alla subroutine desiderata. In particolare:
la subroutine WRITE (350 - 470) crea la "guida" (370) e la muove sulla griglia. Si noti che le istruzioni che muovono "fisicamente" lo sprite, a seconda del tasto premuto, sono soltanto due, la 460 e la 470 e ciò, oltre a risparmiare spazio, velocizza di molto il programma. Le altre istruzioni, invece, provocano l'effetto desiderato (accensione o spegnimento di pixel), a seconda del modo scelto.
La subroutine DISPLAY (490 - 540) calcola, con quattro chiamate del sottoprogramma LEGGI, l'identificatore della sagoma presente sulla griglia.
La subroutine STRINGA (560 - 640) accetta, sulla destra dello schermo, l'identificatore che inserite e visualizza la sagoma relativa, chiamando il sottoprogramma SCRIVI.
La subroutine NEW (660) cancella la sagoma esistente sulla griglia. Inoltre sono più volte utilizzate, nel corso del programma, le subroutine comprese tra le seguenti righe:
680 - 750, che mostra i caratteri costruiti sulla griglia nel colore desiderato. I codici dei caratteri usati sono 128, 129; 130,131. Nella riga 720, tra le virgolette, vanno inseriti i tasti di controllo relativi ai suddetti codici, e quindi le due istruzioni che mostrano il carattere a grandezza normale vanno scritte come segue:
DISPLAY AT(22,25): "(CTRL ,)"; "(CTRL B)"
DISPLAY AT(23,25): "(CTRL A)";"(CTRL C)"
770 - 780, che stampa la "cornice dello schermo".

E per finire....

Il programma non è molto lungo (circa 5 Kbyte), ma si rivelerà sicuramente utile a quanti provvederanno a digitarlo. Ho fatto particolare attenzione a costruire il listato nel modo più strutturato possibile, anche perché il BASIC esteso facilita l'uso di simili tecniche. Quindi ogni modifica che vorrete eventualmente apportare dovrebbe risultare agevole. La strada migliore per impadronirsi di un linguaggio, infatti, è quella di usarlo il più possibile, sia scrivendo propri programmi che modificando, secondo le proprie esigenze, quelli degli altri e l'Extended BASIC non sfugge di certo a questa regola. Buon lavoro! ■

PERSONAL SOFTWARE

Uso delle tabelle con TI99/4A

Sono di uso molto frequente nei programmi ad orientamento commerciale

di *Sergio Borsani*

Introduzione

Contro l'opinione dominante che vede in un home computer uno strumento per il tempo libero ed il gioco, le domande che più di frequente mi sono state rivolte riguardano soprattutto applicazioni di tipo commerciale oltre che di natura didattico-scientifica. Molte persone conducono attività che per le limitate dimensioni non giustificano l'acquisto di un personal da sette o otto milioni, oppure, prima di una spesa impegnativa, vogliono capire quali possono essere le prestazioni di un computer e quale la sua attività. In pratica vogliono sperimentare su piccola scala alcune applicazioni per acquisire gli elementi che potranno giustificare la scelta futura. Il programma del listato 1 si inserisce nell'area della gestione di magazzino ed è soltanto una parte semplificata di un programma completo che assolve il solo compito di riepilogare gli articoli venduti e determinarne il valore parziale e totale.

Con ogni probabilità l'argomento interessa non solo i possessori del TI99/4A ma chiunque abbia un piccolo computer. Ebbene, ho usato il TI BASIC invece dell'Extended BASIC proprio per rendere più semplice la sua comprensione e più facile la sua conversione in un'altra versione di BASIC. Nella figura 6 vengono indicate le modifiche necessarie per rendere il programma compatibile con i computer VIC 20 e Commodore 64. La memoria minima utilizzata non supera i 3 Kbyte. Ben più importante a questi fini è lo spazio occupato dalla tabella. Il TI99/4A, senza espansione di memoria può contenere una tabella di 500 x 2 elementi, cioè, ad esempio, 500 numeri di codice e i rispettivi 500 prezzi unitari; con l'espansione di memoria non si riesce, come speravo, a raddoppiare tale numero ma si arriva a gestire una tabella di 900 x 2 elementi anche se rimane ancora molta memoria disponibile per il programma.

Che cos'è una tabella

Un semplice diagramma di flusso può rappresentare un file-archivio, un file-dati e l'elaborazione di un file-archivio aggiornato. Un altro schema altrettanto elementare prevede un file-dati e l'elaborazione di un riepilogo con la consultazione di una tabella. Questa è una struttura formata da variabili con indice e viene altrimenti chiamata matrice, o vettore, quando la matrice sia ad una dimensione. Non va quindi confusa con i "tabelloni elettronici" (spreadsheet) dei programmi come il VisiCalc o il Multiplan. L'uso delle variabili con indice e delle matrici costituisce una svolta significativa per quanti coltivano l'hobby della programmazione ed anche per chi alla programmazione si accosta per motivi professionali. Generalmente accade che dopo aver imparato le istruzioni più comuni di assegnazione, scrittura, salto condizionato e incondizionato, il primo vero scoglio nel quale ci si imbatte è costituito dalle matrici; una volta superato, tuttavia, i programmi acquistano una maggiore complessità e diventano più potenti.

Una tabella va intesa come un file residente in memoria centrale i cui elementi vengono consultati durante l'elaborazione di un altro file residente su nastro. Ovviamente potrebbe risiedere su un dischetto ma, se è possibile, preferisco riferire gli esempi alla configurazione di sistema più diffusa, cioè: console, televisore e registratore a cassette. In questo programma la tabella da consultare contiene i codici di un certo numero di articoli ed i relativi prezzi unitari. Poiché sarebbe assurdo dover scrivere questi dati ogni volta

Uso delle tabelle con TI99/4A

che si avvia il programma, anche la tabella va memorizzata una volta per tutte su nastro e richiamata prima di ogni elaborazione per la quale sia richiesta. Quando la tabella è stata caricata nella memoria centrale si può eseguire il programma principale il quale può valorizzare un certo numero di articoli movimentati ricavando la quantità da un file sequenziale e desumendo il prezzo unitario dalla tabella precedentemente caricata. In questo caso specifico l'output è dato sul video, ma chi possiede una stampante può, con l'aggiunta di poche istruzioni, ottenere una copia permanente dell'elaborato su carta. Chi utilizza contemporaneamente due registratori, può utilizzarne uno in lettura ed uno per salvare i risultati dell'elaborazione. Inoltre chi volesse utilizzare il programma con un numero di record elevato dovrà introdurre un'istruzione CALL KEY per interrompere lo scrolling verso l'alto e per proseguire dopo la lettura di una "videata" senza dover ogni volta interrompere il programma con il comando FCTN (4) e farlo riprendere con il comando CON (ENTER).

Come funziona il programma

Il programma si apre con il menu riportato nella figura 1. I punti 1, 2, 3 e 4 servono per inizializzare i file che verranno utilizzati dal programma principale e per controllare che il loro contenuto sia corretto. L'elaborazione avviene al punto 5 (STAMPA RIEPILOGO) e prevede necessariamente l'esistenza dei due file (file Tabella e file Movimenti) creati selezionando da menu i punti 1 e 3. Si scelga quindi il punto 1. Compariranno numeri progressivi e la richiesta di numero di codice e prezzo unitario. Ci si limiti a numeri

```
1. CREA TABELLA
2. STAMPA TABELLA
3. CREA FILE MOVIMENTI
4. STAMPA FILE MOVIMENTI
5. STAMPA RIEPILOGO
6. FINE PROGRAMMA
```

Figura 1. *Il menu in apertura di programma. È necessaria l'inizializzazione del file Tabella e del file Movimenti per l'elaborazione del riepilogo.*

```
TABELLA

COD.: 1001      PREZZO  12000
COD.: 1002      PREZZO    750
COD.: 1003      PREZZO   1500
COD.: 1004      PREZZO   8400
COD.: 1005      PREZZO   6000

 premi space per continuare
```

Figura 2. *Selezionando il tasto 2 si ottiene la stampa del contenuto della tabella, nel caso specifico i numeri di codice degli articoli ed i prezzi unitari. Non è necessario che i codici siano progressivi ma devono essere ordinati in ordine crescente. In configurazione base il TI99/4A può gestire 500 articoli.*

di codice con non più di quattro o cinque caratteri altrimenti in seguito sorgeranno problemi di tabulazione sul limitato display a 28 colonne del nostro computer. Ed ancora, la matrice che costituirà la tabella è stata dimensionata con un massimo di 100 elementi; l'introduzione di un numero maggiore a 100 elementi di tabella provocherebbe in seguito la comparsa del messaggio d'errore BAD SUBSCRIPT per il riferimento ad un indice superiore a quello ammesso dal dimensionamento. A titolo esemplificativo si introducano i dati riportati nella figura 2. I numeri di codice vanno dal 1001 al 1005 ed i rispettivi prezzi unitari sono

```
FILE MOVIMENTI
CODICE: 1002  QUANTITA' 34
CODICE: 1002  QUANTITA'  6
CODICE: 1002  QUANTITA'  3
CODICE: 1004  QUANTITA' 12
CODICE: 1004  QUANTITA'  6
CODICE: 1005  QUANTITA'  3
CODICE: 1005  QUANTITA' 10
CODICE: 1005  QUANTITA'  5
CODICE: 1005  QUANTITA'  6
 premi space per continuare
```

Figura 3. *Display nella fase di controllo del file Movimenti.*
Più record possono riguardare lo stesso articolo e, mancando una routine di ordinamento, si avrà l'attenzione di scrivere i numeri di codice in ordine crescente.

```
          RIEPILOGO

CODICE   QTA   PREZZO    VALORE
-----------------------------
 1002     43     750      32250
 1004     18    8400     151200
 1005     24    6000     144000
-----------------------------
             TOTALE     327450
-----------------------------

 premi space per continuare
```

Figura 4. *Come appare l'elaborato finale. Tutti i movimenti riguardanti lo stesso articolo vengono presentati sotto un'unica voce comprendente il numero di codice dell'articolo, la quantità (QTA), il prezzo unitario ed il valore complessivo.*
Il prezzo unitario viene desunto dal programma stesso con la consultazione di una tabella caricata in memoria nella fase iniziale.

12000, 750, ecc. Per indicare al computer che si è terminato bisogna scrivere FINE alla successiva richiesta di codice. Naturalmente, durante questa prima fase di inizializzazione della tabella, è necessario seguire le indicazioni che appaiono sul video

Uso delle tabelle con TI99/4A

per l'uso del registratore. È altresì importante segnare le posizioni indicate dal contanastro (tape counter) per poter ritrovare inequivocabilmente il file appena registrato e non sovrapporre a questo altre registrazioni. Ritornati al menu si potrà controllare il contenuto del file Tabella selezionando il numero 2. Sul video si vedrà quanto è illustrato dalla figura 2. Premendo poi la barra spaziatrice (o qualsiasi altro tasto) si ritorna al menu principale per passare alla fase successiva: la creazione del file Movimenti. Questo potrebbe rappresentare gli articoli venduti nel corso della giornata, ogni suo record contiene un numero di codice e la quantità. È anche naturale presupporre che lo stesso articolo sia stato venduto più volte e che pertanto nel file Movimenti più record contengano lo stesso numero di codice. Inoltre, non tutti gli articoli della tabella verranno necessariamente movimentati, ma tutti gli articoli movimentati dovranno avere i codici presenti in tabella.

Importante ai fini del corretto funzionamento del programma è che i numeri di codice siano in ordine crescente! Se si desidera creare un file Movimenti con i codici non ordinati sarà necessario aggiungere una routine per l'ordinamento in modo da operare sempre con file ordinati per numero crescente di codice. Per esempio, i dati potrebbero essere desunti dalla figura 3.

Mentre il file Tabella viene caricato in memoria centrale e deve sottostare ai limiti imposti dal dimensionamento, il file Movimenti è condizionato solo dalla lunghezza del nastro magnetico in quanto i suoi record vengono letti ed elaborati uno alla volta. La figura 3 riporta il display durante la fase di controllo e stampa del file Movimenti (opzione 3 del menu). Anche in questo caso non si dimentichi di segnare i valori indicati dal contanastro. L'elaborazione consiste nel caricare in memoria la tabella, leggere ad uno ad uno gli articoli movimentati, sommare le quantità che si riferiscono ad uno stesso articolo, ricavare dalla tabella il prezzo unitario, calcolare il valore (quantità x prezzo unitario) per ogni articolo ed infine stampare il valore totale. L'utente, selezionato il punto 5 del menu, dovrà solo seguire le indicazioni relative all'uso del registratore, il riepilogo apparirà come nella figura 4.

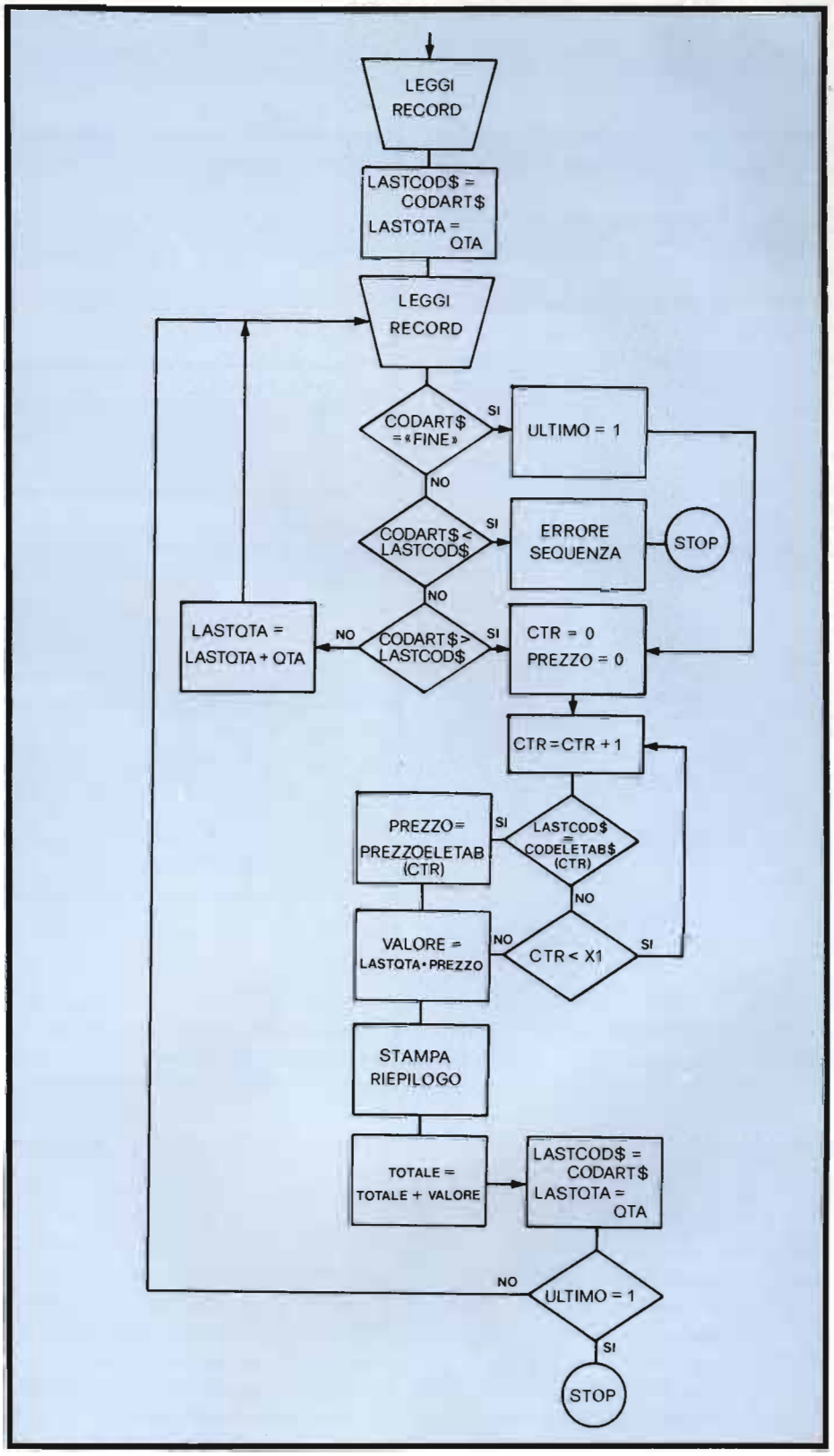


Figura 5. *Diagramma a blocchi riguardante la parte principale del programma. I record del file Movimenti vengono letti ed elaborati uno alla volta sequenzialmente ed i loro codici confrontati con quelli presenti in tabella per ricavare i prezzi unitari.*

Uso delle tabelle con TI99/4A

Come il programma utilizza la tabella

Caricata la tabella in memoria, l'elaborazione del riepilogo inizia alla riga 1120 del programma. Viene letto il primo record e si salvano il numero di codice e la quantità. Si legge quindi il secondo record. Se il codice contiene la parola FINE si elabora l'unico dato letto ed il programma termina. Se il nuovo numero di codice è inferiore al precedente viene inviato il messaggio ERRORE SEQUENZA ed il programma termina. Se il numero di codice è lo stesso, si somma la nuova quantità alla precedente e si legge il record successivo. Infine, se l'ultimo codice letto è maggiore del precedente significa che c'è stata una "rottura di codice" e prima di leggere un altro record bisogna calcolare e stampare il valore relativo al primo articolo. In questo caso il programma passa alla linea 1250 dove inizia la ricerca nella tabella del prezzo unitario. La variabile CTR (contatore) costituisce l'indice dell'elemento della tabella. Si confronta il codice dell'articolo movimentato con il primo codice presente in tabella, poi con il secondo, con il terzo, e così via (linea 1280 del programma). Se i due codici risultano uguali il prezzo unitario sarà l'elemento di tabella PREZZOELETAB (CTR) con lo stesso indice.
Se un codice del file Movimenti non c'è in tabella, il prezzo unitario viene posto uguale a zero e la elaborazione continua. La figura 5 mostra il diagramma a blocchi relativo a questa parte del programma.

Descrizione del listato

100-240 Menu e selezione opzione.
250-380 Creazione e registrazione della tabella su nastro. la particolare funzione TAB di linea 300 serve ad allineare i numeri a destra.
390-540 Lettura e stampa della tabella per controllarne il contenuto, soprattutto l'ordine crescente dei numeri di codice.
550-700 Creazione file movimenti. La fase termina quando si digita FINE alla richiesta del codice.
710-860 Stampa del file Movimenti per il controllo prima dell'elaborazione principale. L'ultimo record contenente la parola FINE non viene stampato (linea 770).
870 Inizia la parte principale per l'elaborazione del riepilogo.
880 Dimensionamento della tabella. Un aumento indiscriminato dell'indice provoca il messaggio d'errore MEMORY FULL e l'arresto del programma.
890-940 Apertura del file ed indicazioni sull'uso del registratore.
950-1010 Caricamento della tabella in memoria. L'ultimo elemento, contenente la parola FINE, non viene ovviamente caricato.
1020-1110 Inizializzazione di alcune variabili, apertura del file movimenti e stampa della testata.
1120-1240 Lettura dei record, confronto tra un codice ed il successivo, rivelazione dell'ultimo record e di errori di sequenza.
1250-1300 Consultazione della tabella per ricavare il prezzo unitario.
1310-1370 Stampa di una riga di riepilogo.
1380-1460 Stampa del totale, chiu-

TAVOLA DI CONVERSIONE

TI BASIC	BASIC COMMODORE 64
CALL CLEAR	PRINT "(CLR/HOME)"
CALL KEY (0, K, S) IF S = 0 THEN ...	GET A$ IF A$ = " " THEN ...
K (K memorizza direttamente il numero di codice ASCII del tasto premuto)	K = ASC (A$)
IF (K < 49) + (K > 54) THEN ... (la funzione OR è disponibile solo in TI EXT. BASIC)	IF K < 49 OR K > 54 THEN ...
OPEN # 4: "CS1", INPUT, SEQUENTIAL,	OPEN 4, 1, 0, "nome del file"
INTERNAL, FIXED OPEN # 4: "CS1", INPUT, SEQUENTIAL, INTERNAL, FIXED	OPEN 4, 1, 1, "nome del file"
CTR (il TI BASIC considera tutti i numeri come reali)	CT% (il BASIC Commodore distingue le variabili numeriche intere)
TOTALE (il TI BASIC accetta nomi di variabili lunghi fino a 15 caratteri, tutti identificati)	T (il BASIC Commodore identifica solo i primi due caratteri delle variabili. TOTALE non si può usare perché viene identificata come la parola riservata TO)
lo stesso dicasi per ... PREZZO COD$ CODELETAB$ (100) PREZZOELETAB (100) LASTCOD$ LASTQTA Display su 28 colonne.	PZ CO$ CT$ (100) PT (100) LC$ LQ Display su 40 colonne. Si possono modificare le funzioni TAB.

Figura 6. *Elementi di confronto per la conversione nel BASIC del VIC 20 e del Commodore 64.*

Uso delle tabelle con TI99/4A

sura del riepilogo e chiusura del file Movimenti.
1470-1480 Fine del programma.

Variabili usate nel programma

K Contiene il numero di codice ASCII del tasto premuto.
CTR Viene usata come contatore in varie fasi.
COD$ Numero di codice (alfanumerico) dell'articolo.
PREZ Prezzo unitario.
CODART$ Codice articolo letto dal file movimenti.
QTA Quantità.
CODELETAB$ () Elemento di tabella consistente nel codice dell'articolo.
PREZZOELETAB () Elemento di tabella consistente nel prezzo unitario.
TOTALE Valore complessivo degli articoli movimentati.
X Contatore degli elementi di tabella.
X1 Numero degli elementi in tabella.
ULTIMO Variabile flag; il suo valore viene posto uguale ad uno quando è stato letto l'ultimo record del file Movimenti.
LASTCOD$ Ultimo codice letto prima dell'attuale.
LASTQTA Ultima quantità letta prima dell'attuale.
PREZZO Prezzo unitario desunto dalla consultazione della tabella.
VALORE Quantità per prezzo unitario. ■

Listato 1. *Il programma per la creazione e l'uso di tabelle.*

```
100 REM   PROVA USO TABELLA
110 REM  ********************

120 REM  versione: TI BASIC
130 CALL CLEAR
140 PRINT "1. CREA TABELLA"
150 PRINT "2. STAMPA TABELLA
"
160 PRINT "3. CREA FILE MOVI
MENTI"
170 PRINT "4. STAMPA FILE MO
VIMENTI"
180 PRINT "5. STAMPA RIEPILO
GO"
190 PRINT "6. FINE PROGRAMMA
"::::
200 CALL KEY(0,K,S)
210 IF S=0 THEN 200
220 IF (K<49)+(K>54)THEN 200
230 K=K-48
240 ON K GOTO 250,390,550,71
0,870,1470
250 CALL CLEAR
260 OPEN #4:"CS1",OUTPUT,SEQ
UENTIAL,INTERNAL,FIXED
270 CALL CLEAR
280 CTR=0
290 CTR=CTR+1
300 PRINT TAB(4-LEN(STR$(CTR
)));CTR;
310 INPUT "CODICE: ":COD$
320 IF COD$="FINE" THEN 340
330 INPUT "     PREZZO: ":PR
EZ
340 PRINT #4:COD$,PREZ
350 PRINT
360 IF COD$<>"FINE" THEN 290
```

Seguito listato 1.

```
370 CLOSE #4
380 GOTO 100
390 CALL CLEAR
400 OPEN #4:"CS1",INPUT ,SEQ
UENTIAL,INTERNAL,FIXED
410 CALL CLEAR
420 PRINT "TABELLA"
430 PRINT
440 INPUT #4:COD$,PREZ
450 IF COD$="FINE" THEN 480
460 PRINT "COD.: ";COD$,"PRE
ZZO:";TAB(28-LEN(STR$(PREZ))
);PREZ
470 GOTO 440
480 PRINT
490 PRINT " premi space per
continuare"
500 CALL KEY(0,K,S)
510 IF S=0 THEN 500
520 CALL CLEAR
530 CLOSE #4
540 GOTO 100
550 CALL CLEAR
560 OPEN #3:"CS1",OUTPUT,SEQ
UENTIAL,INTERNAL,FIXED
570 CALL CLEAR
580 PRINT "MOVIMENTI"
590 PRINT
600 CTR=0
610 CTR=CTR+1
620 PRINT TAB(4-LEN(STR$(CTR
)));CTR;
630 INPUT "CODICE: ":CODART$
640 IF CODART$="FINE" THEN 670
650 INPUT "  QUANTITA': " :QTA
660 PRINT
670 PRINT #3:CODART$,QTA
680 IF CODART$="FINE" THEN 6
```

Uso delle tabelle con TI99/4A

Seguito listato 1.

```
90 ELSE 610
690 CLOSE #3
700 GOTO 100
710 CALL CLEAR
720 OPEN #3:"CS1",INPUT ,SEQ
UENTIAL,INTERNAL,FIXED
730 CALL CLEAR
740 PRINT "FILE MOVIMENTI"
750 PRINT
760 INPUT #3:CODART$,QTA
770 IF CODART$="FINE" THEN 8
00
780 PRINT "CODICE: ";CODART$
,"QUANTITA':";TAB(29-LEN(STR
$(QTA)));STR$(QTA)
790 GOTO 760
800 PRINT
810 PRINT " premi space per
continuare"
820 CALL KEY(0,K,S)
830 IF S=0 THEN 820
840 CALL CLEAR
850 CLOSE #3
860 GOTO 100
870 CALL CLEAR
880 DIM CODELETAB$(100),PREZ
ZOELETAB(100)
890 PRINT "* CARICA FILE TAB
ELLA"
900 OPEN #3:"CS1",INPUT ,SEQ
UENTIAL,INTERNAL,FIXED
910 TOTALE=0
920 X=0
930 PRINT
940 PRINT "* RUNNING ..."
950 INPUT #3:COD$,PREZ
960 IF COD$="FINE" THEN 1010
970 X=X+1
980 CODELETAB$(X)=COD$
990 PREZZOELETAB(X)=PREZ
1000 GOTO 950
1010 CLOSE #3
1020 X1=X
1030 ULTIMO=0
1040 PRINT
1050 PRINT "* CARICA FILE MO
VIMENTI"
1060 OPEN #5:"CS1",INPUT ,SE
QUENTIAL,INTERNAL,FIXED
1070 CALL CLEAR
1080 PRINT TAB(10);"RIEPILOGO"
```

Seguito listato 1.

```
1090 PRINT ::
1100 PRINT "CODICE  QTA  PRE
ZZO   VALORE"
1110 PRINT "-----------------
-------------"
1120 INPUT #5:CODART$,QTA
1130 LASTCOD$=CODART$
1140 LASTQTA=QTA
1150 INPUT #5:CODART$,QTA
1160 IF CODART$<>"FINE" THEN
 1190
1170 ULTIMO=1
1180 GOTO 1250
1190 IF CODART$>=LASTCOD$ TH
EN 1220
1200 PRINT TAB(6);"ERRORE SE
QUENZA"
1210 GOTO 1450
1220 IF CODART$>LASTCOD$ THE
N 1250
1230 LASTQTA=LASTQTA+QTA
1240 GOTO 1150
1250 CTR=0
1260 PREZZO=0
1270 CTR=CTR+1
1280 IF LASTCOD$=CODELETAB$(
CTR)THEN 1300
1290 IF CTR<X1 THEN 1270 ELS
E 1310
1300 PREZZO=PREZZOELETAB(CTR
)
1310 VALORE=LASTQTA*PREZZO
1320 PRINT LASTCOD$;TAB(11-L
EN(STR$(LASTQTA)));LASTQTA;T
AB(18-LEN(STR$(PREZZO)))
;PREZZO;
1330 PRINT TAB(29-LEN(STR$(V
ALORE)));STR$(VALORE)
1340 TOTALE=TOTALE+VALORE
1350 LASTQTA=QTA
1360 LASTCOD$=CODART$
1370 IF ULTIMO<>1 THEN 1150
1380 PRINT "-----------------
-------------"
1390 PRINT TAB(14);"TOTALE";
TAB(29-LEN(STR$(TOTALE)));ST
R$(TOTALE)
1400 PRINT "-----------------
-------------"
1410 PRINT " premi space per
 continuare"
1420 CALL KEY(0,K,S)
1430 IF S=0 THEN 1420
1440 CALL CLEAR
1450 CLOSE #5
1460 GOTO 100
1470 CALL CLEAR
1480 END
```

alla "N" il 94, che non hanno proprio niente in comune con i numeri del codice ASCII.
Un'ultima possibilità che si deve considerare, prima di eseguire il programma-sonda, è la sfortunata coincidenza di trovare nella ROM tre byte consecutivi uguali ai nostri tre numeri, ma che vi corrispondono solo casualmente. Tuttavia questa probabilità è estremamente bassa e, in ogni caso, ci si accorgerebbe molto facilmente dell'errore osservando che i numeri successivi non corrispondono ad altre istruzioni.

Vediamo finalmente che cosa succede a far girare il nostro primo programma: dopo qualche minuto di attesa si udiranno tre beep ed il display visualizzerà il numero 17173. Abbiamo così trovato quello che stavamo cercando, ma dobbiamo fare ancora una cosa, cioè delimitare con precisione l'area del buffer, perché non sappiamo in che posizione sia posta l'istruzione RUN rispetto a questo. Infatti niente deve farci ritenere che il comando RUN sia posto all'inizio del buffer oppure alla fine.
Per circoscrivere la zona di memoria che ci interessa utilizzeremo il programma 2, che funziona nel modo seguente: dato un certo indirizzo iniziale, il computer cerca di interpretare come lettere i byte corrispondenti a quell'indirizzo, ed ai seguenti, purché siano numeri compresi fra 81 e 106, che sappiamo corrispondere alle lettere dell'alfabeto.
Dando come indirizzo iniziale il nostro 17173, comparirà sul display una lunga sequenza di lettere e numeri che inizierà così:
R U N 176 I 122 198 R E T U R N 222 H 30 164 R E A D. Vedremo così sfilare sotto i nostri occhi una buona parte delle istruzioni dello Sharp, fino a quando non inizieranno a comparire lettere e numeri senza senso, segno indiscutibile della fine del buffer. Per trovarne l'inizio dovremo dare invece degli indirizzi di partenza inferiori a 17173. Con qualche tentativo si potrà verificare che l'indirizzo iniziale è 16683, mentre quello finale è 17377.

La prossima volta commenteremo lungamente queste nuove scoperte e vedremo che si riveleranno estremamente utili; per il momento mi limito a fare qualche piccola osservazione. Innanzitutto, come era da aspettarsi, troviamo normalmente elencate in questa lunga lista anche tutte quelle istruzioni stranissime che avevamo scoperto in precedenza.
Cosa abbastanza curiosa è che siano disposte in una specie di ordine alfabetico, forse per permettere al sistema operativo un confronto più rapido. Infine, per chiudere con un po' di suspence, fra una istruzione e la successiva si trovano quattro numeri "misteriosi", il cui segreto verrà svelato, almeno in parte, la prossima volta e ci permetteranno di giungere fino al "cuore" di questo computer.

TEXAS TI99/4A

Gestione file con un solo registratore

di Sergio Borsani

Uno dei compiti principali ai quali può essere adibito un elaboratore è quello di memorizzare e tenere aggiornati dei dati, utilizzando le così dette "memorie di massa", cioè unità di registrazione su nastro o su disco. Le prime hanno l'unico vantaggio di essere molto più economiche; le seconde presentano la caratteristica di poter accedere direttamente ad un dato e sono più veloci.
Spesso mi sono sentito rivolgere la domanda su cosa sia possibile fare nei riguardi della gestione dei file, nella configurazione base, con un solo registratore. Francamente ritengo che le possibilità siano limitate. Con due unità di registrazione le cose andrebbero appena un po' meglio ed è per questa ragione che viene fornito un cavetto con doppi attacchi, uno per lettura/scrittura ed un altro per sola lettura (ho saputo recentemente che gli ultimi cavetti vengono forniti con un'unica uscita).

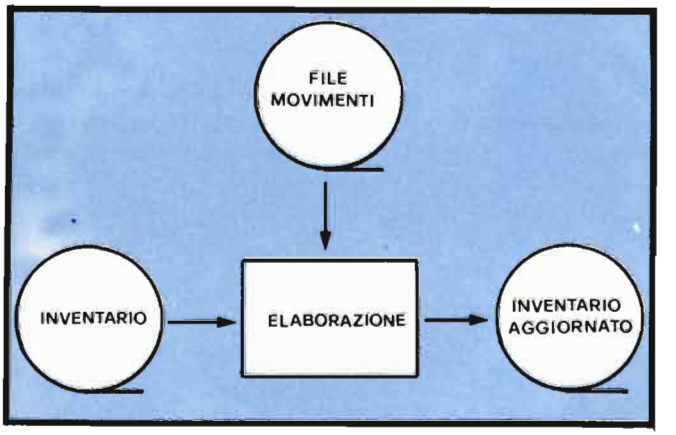


Figura 1. *Flow-chart relativo ad una elaborazione di file sequenziali. I simboli circolari indicano che essi risiedono su nastro magnetico.*

Lo schema principale, e semplificato, in questo genere di elaborazione è illustrato dal flow-chart di figura 1.
È evidente che l'unica via da seguire per chi dispone di un unico registratore è quella di memorizzare interamente il file da aggiornare nella memoria interna del computer (RAM), aggiornare tale file utilizzandone un secondo residente su nastro e quindi, terminate le operazioni di aggiornamento, salvare il file su una porzione libera del nastro, che sia ben identificabile con il numero del tape counter.

I SEGRETI DEI PERSONAL

Gestione file con un solo registratore

```
100 A$="PROVA RECORD"
110 OPEN #3:"CS1",OUTPUT,SEQUENTIAL,FIXED
120 FOR J=1 TO 10
130 PRINT #3:A$
140 NEXT J
150 CLOSE #3
160 DIM R$(10)
170 OPEN #3:"CS1",INPUT ,SEQUENTIAL,FIXED
180 N=0
190 FOR J=1 TO 10
200 INPUT #3:A$
210 N=N+1
220 R$(N)=A$
230 NEXT J
240 CLOSE #3
250 CALL CLEAR
260 FOR J=1 TO 10
270 PRINT R$(J)
280 NEXT J
290 END
```

Listato 1. *Semplice programma utilizzato per scrivere e leggere un file su nastro.*

```
100 A$="PROVA RECORD"
110 OPEN #3:"CS1",OUTPUT,SEQUENTIAL,FIXED
120 FOR J=1 TO 10
130 PRINT #3:A$;J
140 NEXT J
150 CLOSE #3
160 DIM R$(10,2)
170 OPEN #3:"CS1",INPUT ,SEQUENTIAL,FIXED
180 N=0
190 FOR J=1 TO 10
200 INPUT #3:A$,NUMREC
210 N=N+1
220 R$(N,1)=A$
225 R$(N,2)=STR$(NUMREC)
230 NEXT J
240 CLOSE #3
250 CALL CLEAR
260 FOR J=1 TO 10
270 PRINT R$(J,1);" ";R$(J,2)
280 NEXT J
290 END
```

Listato 2. *Lo stesso programma del listato 1, ma con il record composto di 2 campi.*

```
110 OPEN #3:"CS1",OUTPUT,SEQUENTIAL,INTERNAL,FIXED

170 OPEN #3:"CS1",INPUT ,SEQUENTIAL,INTERNAL,FIXED
```

Listato 3. *Linee da modificare per rendere funzionante il programma 2.*

```
100 REM   FILE MOVIMENTI
110 REM ******************
120 CALL CLEAR
130 N=0
140 OPEN #3:"CS1",OUTPUT,SEQUENTIAL,
    INTERNAL,FIXED
150 CALL CLEAR
160 DISPLAY AT(6,1):"RECORD N."
170 DISPLAY AT(11,1):"---- - --------"
180 DISPLAY AT(12,1):"COD. T QUANTITA'"
190 N=N+1
200 DISPLAY AT(6,10):N
210 CALL HCHAR(10,3,32,28)
220 ACCEPT AT(10,1)SIZE(4):COD$
230 IF COD$="FINE" THEN 260
240 ACCEPT AT(10,6)VALIDATE("EU")SIZE(1):TM$
250 ACCEPT AT(10,8)VALIDATE(DIGIT)SIZE(8):QTA
260 PRINT #3:COD$;TM$;QTA
270 IF COD$<>"FINE" THEN 190
280 CLOSE #3 :: CALL CLEAR :: END
```

Listato 4. *Programma per creare il file Movimenti.*

Il manuale è effettivamente povero di esempi in questo campo, e prima di passare a programmi più complessi è preferibile eseguire alcune esperienze elementari.

L'elemento principale di un file dati è il record. Un file è formato dalla successione di tanti record. Nel listato 1 i record sono tutti uguali e sono rappresentati dalla stringa A$.

Il numero che accompagna l'istruzione OPEN di linea 110 è a discrezione del programmatore, ma deve corrispondere alle successive PRINT #. "CS1" si riferisce all'unità di registrazione; OUTPUT indica che i dati vengono trasferiti dal computer al nastro; inoltre, l'organizzazione può essere solo sequenziale (SEQUENTIAL) ed i record devono essere tutti della stessa lunghezza (FIXED).

Nella seconda parte del programma (linee 160-290) il file viene letto e caricato nella matrice unidimensionale, o variabile con indice, R$ e stampato sul video. Il record, normalmente, non contiene una sola informazione, ma è una struttura che viene suddivisa in più campi, contenenti informazioni elementari tra di loro omogenee.

I campi di un record potrebbero contenere, ad esempio, nome, indirizzo e numero di telefono, oppure il numero di codice, la descrizione di un articolo ed il prezzo unitario, ecc.

Modifichiamo il listato n. 1 alle linee 130, 160, 200, 270 ed aggiungiamo la riga 225. Ora il record è formato da due campi: A$ e J nella linea 130 (listato 2).

Se lanciate il programma, tuttavia, vi accorgerete che c'è qualcosa che non va.

Si potrebbe rimediare aggiungendo una routine che trasformi il campo numerico in variabile stringa, porti i due campi ad una determinata lunghezza, aggiungendo space (CHR$ (32)), ed infine sommi i due campi in modo da ottenere un'unica stringa alfanumerica.

In fase di lettura si dovrebbe decodificare la stringa sapendo, ad esempio, che i primi 20 caratteri formano il primo campo e che i restanti 8 formano il

I SEGRETI DEI PERSONAL

Gestione file con un solo registratore

```
100 REM   FILE GIACENZE MAGAZZINO
110 REM ***************************
120 CALL CLEAR
130 PRINT TAB(6);"GIACENZE MAGAZZINO":"":"":"":""
140 PRINT "PREMI:":"1. CREA FILE":"2. DISPLAY FILE"
150 PRINT "3. RICHIAMA FILE":"4. AGGIORNA FILE"
160 PRINT "5. REGISTRA FILE":"6. FINE PROGRAMMA":""
170 CALL KEY(0,K,S):: IF S=0 THEN 170
180 IF K<49 OR K>54 THEN 170
190 K=K-48
200 ON K GOTO 210,500,400,590,710,800
210 CALL CLEAR
220 N=0
230 OPEN #3:"CS1",OUTPUT,SEQUENTIAL,INTERNAL,FIXED
240 CALL CLEAR
250 DISPLAY AT(6,1):"RECORD N."
260 DISPLAY AT(11,1):"---- ------------   --------"
270 DISPLAY AT(12,1):"COD. ARTICOLO        GIACENZA"
280 N=N+1
290 DISPLAY AT(6,10):N
300 CALL HCHAR(10,3,32,28)
310 ACCEPT AT(10,1)SIZE(4):COD$
320 IF COD$="FINE" THEN 350
330 ACCEPT AT(10,6)SIZE(12):ART$
340 ACCEPT AT(10,21)VALIDATE(DIGIT)SIZE(8):QTA
350 PRINT #3:COD$
360 IF COD$="FINE" THEN 390
370 PRINT #3:ART$,STR$(QTA)
380 GOTO 280
390 CALL CLEAR :: CLOSE #3 :: GOTO 120
400 CALL CLEAR
410 DIM CAMP$(100,3)
420 OPEN #3:"CS1",INPUT ,SEQUENTIAL,INTERNAL,FIXED
430 CTR=0
440 CTR=CTR+1
450 INPUT #3:CAMP$(CTR,1)
460 IF CAMP$(CTR,1)="FINE" THEN 490
470 INPUT #3:CAMP$(CTR,2),CAMP$(CTR,3)
480 GOTO 440
490 CLOSE #3 :: CALL CLEAR :: GOTO 120
500 CALL CLEAR :: FOR J=1 TO CTR-1
510 PRINT "* RECORD";J;"*"
520 PRINT CAMP$(J,1);TAB(6);CAMP$(J,2);TAB(20);
530 PRINT USING "########":VAL(CAMP$(J,3))
540 PRINT :: NEXT J
550 PRINT
560 PRINT TAB(10);"FINE FILE"
570 PRINT TAB(8);"PREMI UN TASTO"
580 CALL KEY(0,K,S):: IF S=0 THEN 580 ELSE 120
590 CALL CLEAR
600 OPEN #4:"CS1",INPUT ,SEQUENTIAL,INTERNAL,FIXED
610 INPUT #4:COD$,TM$,QTA
620 IF COD$="FINE" THEN 700
630 FOR J=1 TO CTR-1
640 IF COD$<>CAMP$(J,1)THEN 680
650 IF TM$="E" THEN W=1 ELSE W=-1
660 GIACENZA=VAL(CAMP$(J,3))
670 CAMP$(J,3)=STR$(GIACENZA+W*QTA):: GOTO 610
680 NEXT J
690 PRINT "ERRORE CODICE MOVIMENTO";J :: GOTO 610
700 CALL CLEAR :: CLOSE #4 :: GOTO 120
710 CALL CLEAR
720 OPEN #5:"CS1",OUTPUT,SEQUENTIAL,INTERNAL,FIXED
730 N=0
740 N=N+1
750 PRINT #5:CAMP$(N,1)
760 IF CAMP$(N,1)="FINE" THEN 790
770 PRINT #5:CAMP$(N,2);CAMP$(N,3)
780 GOTO 740
790 CALL CLEAR :: CLOSE #5 :: GOTO 120
800 CALL CLEAR :: END
```

Listato 5. *Programma principale che elabora l'inventario aggiornato.*

secondo campo.

Alcuni linguaggi di tipo commerciale, come il COBOL, sono particolarmente adatti alla definizione dei campi e dei record. Ma non è detto che con il BASIC le cose debbano andare poi tanto male. Provate a sostituire le OPEN del listato 2 con quelle del listato 3.

Il manuale dà solo brevi accenni sull'uso dell'opzione INTERNAL, contrapposta all'altra che ha valore di default, DISPLAY. Se fate girare il programma vi convincerete che il formato INTERNAL è quello che fa per noi, in questa circostanza.

Ora che sappiamo registrare e caricare un file, con record formati da più campi, siamo in grado di affrontare un programma più complesso. Immaginiamo di voler aggiornare un file con le giacenze di magazzino (file Inventario), utilizzando i dati di un secondo file (file Movimenti).

Il record dell'inventario è formato da tre campi: codice, descrizione articolo e quantità in giacenza; il record movimento è anch'esso formato da tre campi: codice, tipo movimento (Entrata/Uscita) e quantità. I numeri di codice dei due tipi di record dovranno corrispondere, anche se non tutti gli articoli dell'inventario saranno interessati da variazioni.

Le operazioni dovranno susseguirsi con il seguente ordine: creare i due file con i due distinti programmi riportati nei listati 4 e 5, caricare in memoria il file con le giacenze di magazzino, utilizzando il programma principale (listato 5) e infine, sempre con lo stesso programma, elaborare l'inventario aggiornato.

La matrice che contiene il file Inventario è stata dimensionata a 100. Tale valore può essere aumentato ed è limitato solo dalle capacità di memoria del computer; è tuttavia evidente che in tal modo non potrà mai essere elaborato un file con un numero elevato di record.

Per finire, alcuni brevi commenti.

Si sarebbe potuto utilizzare il TI BASIC, ma si è preferito il TI Extended BASIC per la presenza delle istruzioni DISPLAY AT ed ACCEPT AT (con le opzioni VALIDATE e SIZE), che permettono di creare una "maschera" per l'introduzione dei dati.

Con il registratore non si può usare l'istruzione EOF (End Of File), pertanto bisogna fare in modo che l'ultimo record sia riconoscibile tramite un campo particolare, di solito il primo. Se esso è alfanumerico, si può occupare con la parola "FINE" (soluzione qui adottata), se invece è numerico si riempie, secondo le circostanze, di zeri o di 9.

Attenzione a non sovrapporre programmi e file dati durante le registrazioni!

PERSONAL SOFTWARE

Grafica avanzata sul TI 99/4A

Utilizziamo gli sprite mediante l'Extended BASIC

di *Filippo Cerulo*

La caratteristica forse più interessante dell'Extended BASIC implementato sul TI 99/4A è la possibilità di controllare da programma, in modo relativamente semplice, sagome in movimento sullo schermo. Il nome di queste sagome è sprite (che tradotto in modo letterale significa folletto), nome che ne denuncia l'origine essenzialmente ludica (nel senso di videogame).
In verità poi ci si accorge subito che uno sprite può essere molto utile anche in applicazioni che col gioco hanno poco a che vedere, ma ciò non toglie che la prima immagine che ci viene in mente è quella dell'astronave che spara agli invasori.

È meglio dire subito che (a parte il nome) i nostri sprite hanno poco in comune con quelli del Commodore 64 e sono fondamentalmente diversi dal Player-Missile implementato sui computer Atari. Ciò soprattutto per una precisa scelta tecnica: niente bit da settare con relative PEEK e POKE, niente mappe di memoria da consultare, niente locazioni misteriose da scoprire, ma solo sottoprogrammi su ROM pronti all'uso e direttamente richiamabili da BASIC, con eventuale passaggio di parametri. Naturalmente, non tutti possono essere d'accordo con questa filosofia, ma mi sembra la scelta più logica, se consideriamo la struttura del computer e soprattutto dell'ottimo interprete BASIC. Il rovescio della medaglia è una certa lentezza di alcune istruzioni, ma niente di irreparabile.
D'altronde non si può avere tutto dalla vita e dal ... personal.

```
200  CALL CLEAR :: CALL SCREEN (15) :: DIM MU (24)
210  CALL MAGNIFY (2) :: CALL CHAR (132, "60F0F8FCF8F06",
     133, "000000F", 131, RPT$ ("F", 16), 129, "0", 130, RPT$ ("0F", 8))
     :: CALL COLOR (13, 7, 1)
220  FOR I = 2 TO 19 :: MU (I) = 2 :: NEXT I
230  K = 1 :: VE = 12 :: KA = 0 :: COL = 20
240  J = 0 :: CALL VCHAR (1, COL-1, 32, 24)
250  FOR I = 2 TO 19 :: IF MU (I) > 0 THEN CALL HCHAR (I, COL,
     129 + MU (I)) :: J = J + 1
260  NEXT I :: IF J = 0 THEN 370
270  CALL SPRITE (# 2, 132, 5, 233, 17) :: IF K > 9 THEN K = K - 10
     :: KA = KA + 1 :: CALL SPRITE (# 4, KA + 48, 5, 169, 65)
280  CALL SPRITE (# 5, K + 48, 5, 169, 81) :: CALL MOTION (# 2,
     VE, 0)
290  CALL KEY (5, COD, ST) :: IF COD <> 32 THEN 350
300  CALL POSITION (# 2, RI, CO) :: IF RI > 157 THEN 350
310  CALL SPRITE (# 3, 133, 5, RI, CO, 0, 90) :: P = INT ((RI + 6)/8)
     + 1
320  FOR J = 1 TO 80 + (COL - 20) * 10 :: NEXT J
330  IF MU (P) > 0 THEN MU (P) = MU (P) - 1 :: CALL HCHAR (P,
     COL 129 + MU (P)) :: CALL SOUND (100, - 5, 3)
340  CALL DEL SPRITE (# 3)
350  CALL COINC (# 2, 190, 17, 40, S) :: IF S = 0 THEN 290
360  CALL DEL SPRITE (# 2) :: COL = COL - 1 * (COL < 28) :: K =
     K + 1 :: VE = VE - 1 * (VE < 18) :: GOTO 240
370  CALL DEL SPRITE (ALL) :: DISPLAY AT (3, 2) ERASE ALL :
     "MURO ABBATTUTO IN"
380  CALL SPRITE (# 2, KA + 48, 5, 41, 33, # 3, K + 47, 5, 41, 49)
390  DISPLAY AT (10, 2) : "TENTATIVI." :: DISPLAY AT (15, 2)
     BEEP : "VUOI GIOCARE ANCORA ? (S/N)"
400  CALL KEY (5, COD, ST) :: IF ST = 0 THEN 400
410  IF COD = 83 THEN CALL CLEAR :: CALL DEL SPRITE (ALL) ::
     GOTO 220
420  CALL CLEAR :: STOP
```

Figura 1. *Il listato del programma Darts.*

Folletti

Il nostro sprite è una entità indipendente dal contesto in cui opera. Fondamentalmente possiede i seguenti attributi: una sagoma, un colore, una posizione, una velocità (se è in movimento). È possibile modificare uno qualsiasi di questi attributi, indipendentemente dagli altri ed in qualunque momento, tramite istruzioni dedicate. Possiamo inoltre rilevare da programma alcune informazioni che lo riguardano: la posizione, la sua distanza da un punto dello schermo o da un altro sprite, la sovrapposizione a punti dello schermo o ad altri sprite. Ogni sprite è

identificato da un numero e nelle istruzioni che lo riguardano si fa riferimento all'angolo in alto a sinistra della sagoma che lo rappresenta.
Per uno sprite lo schermo è formato da 256 righe e 256 colonne (indicate nel manuale col prefisso "dot"), ma solo le prime 193 righe sono visualizzabili.

L'istruzione per far nascere uno sprite è:
• CALL SPRITE (# numero di sprite, codice di carattere, colore, riga, colonna, [velocità di riga, velocità di colonna]);
• il "numero di sprite" è un intero tra 1 e 28 compresi, usato come identificatore. In caso di coincidenza sullo schermo, lo sprite di numero inferiore si sovrappone all'altro;
• il "codice di carattere" è il codice ASCII del carattere di cui vogliamo che lo sprite assuma la sagoma, cioè un intero tra 32 e 143;
• il colore è il codice del colore che lo sprite dovrà assumere (un intero tra 1 e 16);
• le "riga, colonna" sono le coordinate della posizione di partenza dello sprite (due interi tra 1 e 256);
• le "velocità di riga, velocità di colonna" sono opzionali e rappresentano l'eventuale movimento da imprimere allo sprite.
(Interi tra − 128 e + 127).

Il seguente mini-programma crea uno sprite con la sagoma del carattere "A" (cod. ASCII 65), di colore blu, che si muove in verticale (fermare l'esecuzione con CLEAR):

```
200 CALL CLEAR
210 CALL SPRITE (# 2, 65, 5, 2, 42, 10, 0)
220 GOTO 220
```

Adesso esamineremo, un po' più da vicino ed uno per uno, gli attributi di uno sprite.

La forma del folletto

Fondamentalmente, per scegliere una sagoma per il folletto si opera ridefinendo caratteri in modo usuale. Uno sprite può avere quattro "grandezze", in corrispondenza a quattro parametri usati nell'istruzione: CALL MAGNIFY (parametro).
In particolare può essere:
a) di un solo carattere (valore di default): CALL MAGNIFY (1);
b) di un solo carattere ingrandito: CALL MAGNIFY (2);
c) di quattro caratteri: CALL MAGNIFY (3);
d) di quattro caratteri ingrandito: CALL MAGNIFY (4).

Nei casi a) e b) la sagoma dello sprite è quella del carattere il cui codice compare nella CALL SPRITE, ingrandita o meno.
Nel seguente programma ridefiniamo il carattere "A" come un ometto che dopo un po' diventa più grande grazie alla istruzione della riga 240 (fermare il programma con CLEAR):

```
200 CALL CLEAR
210 CALL CHAR (65, "1818FF3C3C242424")
220 CALL SPRITE (# 2, 65, 5, 2, 42, 10, 0)
230 FOR I = 2 TO 1000 :: NEXT I
240 CALL MAGNIFY (2)
250 GOTO 250
```

Con una CALL MAGNIFY (2) la sagoma dello sprite rimane invariata, ma le sue dimensioni raddoppiano, cioè lo spazio occupato è quello corrispondente a quattro caratteri.

Nei casi c) e d), invece, la sagoma dello sprite è formata dall'unione di quelle di quattro caratteri. In particolare dal carattere specificato nella CALL SPRITE e dai tre caratteri seguenti, se il codice del primo carattere (quello specificato) è divisibile per quattro. In caso contrario la "conta" comincia col primo carattere, precedente il codice specificato, il cui codice sia divisibile per quattro. La corrispondenza delle sagome procede dall'alto in basso e da destra a sinistra.
Ecco un esempio (fermare il programma con CLEAR):

```
200 CALL CLEAR
210 CALL MAGNIFY (3)
220 CALL SPRITE (# 2, 68, 5, 2, 42, 9, 0)
230 FOR I = 2 TO 2000 :: NEXT I
240 CALL MAGNIFY (4)
250 GOTO 250
```

Nel programma precedente il codice di carattere specificato è 68 ("D"), che è divisibile per quattro. Lo sprite è formato, in sequenza, dalle lettere D E F G (codici 68, 69, 70, 71).
Per quanto detto prima, sostituendo nell'istruzione 220 il codice 68 con, ad esempio, 78 ("N") lo sprite sarebbe formato dalle lettere L M N O (codici 76, 77, 78, 79).

Risulta adesso chiaro che per sagome molto elaborate la parte più noiosa è quella relativa al calcolo degli identificatori esadecimali che devono ridefinire i caratteri. Fortunatamente esistono in commercio programmi chiamati Sprite Editor) che sollevano da questa incombenza rendendo anzi questa fase addirittura divertente. Nel prossimo numero di **Personal Software** presenteremo proprio un potente Sprite Editor, che potrete utilizzare per definire i vostri caratteri. Per ultimo diciamo che si può modificare la sagoma di uno sprite da programma, senza toccare le altre caratteristiche, con l'istruzione:
CALL PATTERN (# numero di sprite, codice di carattere)
dove il "numero di sprite" è quello dello sprite che vogliamo modificare ed il "codice di carattere", secondo quanto detto prima, è un intero tra 32 e 143 che si riferisce alla nuova sagoma. Provate ad aggiungere le seguenti linee al programma precedente ed osservate i risultati.

```
245 FOR I = 2 TO 2000 :: NEXT I
247 CALL PATTERN (# 2, 84)
```

Il colore del folletto

I colori disponibili sul TI 99/4A sono sedici, identificati da numeri interi a partire da uno. Di solito, di un insieme di caratteri si può sceglie-

re sia il colore del carattere (foreground) sia quello dello sfondo (background). Il colore standard dello sfondo per uno sprite è "trasparente" (cod. = 1) e non può essere modificato. Possiamo invece scegliere il colore del folletto attraverso il codice-colore appropriato nella CALL SPRITE e modificarlo in qualunque momento con l'istruzione:
CALL COLOR (# numero di sprite, codice-colore)

La posizione del folletto

Lo schermo del 99 è formato da 24 righe e 32 colonne. Dato che ogni carattere viene formato su una griglia di 8 x 8 pixel, l'intero schermo comprende 192 x 256 pixel. Quando ci riferiamo ad uno sprite, la posizione sullo schermo è riferita al pixel; il video è formato da 256 pi-righe (di cui le prime 192 visibili) e da 256 pi-colonne. Le righe sono numerate, a partire da uno, dall'alto in basso, le colonne da destra a sinistra. Ad esempio, se vogliamo posizionare uno sprite sulla riga 15 e sulla colonna 6, dovremo specificare una pi-riga uguale a 113 ed una pi-colonna uguale a 41. La relazione è infatti:

pi-riga = (riga ★ 8) − 7

e ciò vale anche per le colonne. La posizione di partenza per un folletto è quella specificata nella CALL SPRITE, riferita all'angolo in alto a sinistra della sagoma che lo definisce.
La posizione può essere modificata con una istruzione del tipo:
CALL LOCATE (# numero di sprite, pi-riga, pi-colonna)

Il breve programma che segue muove a caso uno sprite sul video, mostrando le coordinate della posizione. Notate che se il numero di pi-riga generato è maggiore di 192, lo sprite non è visibile (fermate il programma con CLEAR).

```
200 CALL CLEAR :: RANDOMIZE
210 CALL MAGNIFY (3) :: CALL SPRITE (# 2, 80, 2, 200, 2)
220 RI = INT (RND ★ 256) + 1 :: CO = INT (RND ★ 256) + 1
230 DISPLAY AT (2, 1) SIZE (12) : "RIGA :"; RI
240 DISPLAY AT (2, 16) BEEP : "COLONNA :"; CO
250 CALL LOCATE (# 2, RI, CO)
260 FOR I = 2 TO 1500 :: NEXT I
270 GOTO 220
```

Il movimento del folletto

Il movimento di uno sprite è il risultato della somma vettoriale di due grandezze: la velocità di riga e la velocità di colonna. Esse sono rappresentate da numeri interi compresi tra − 128 e + 127. Se assumiamo l'origine di un piano cartesiano in alto a sinistra dello schermo, con l'asse delle ascisse orientato verso il basso e quello delle ordinate da sinistra verso destra, la velocità di riga è la componente della velocità secondo l'asse "x" e la velocità di colonna la componente secondo l'asse "y". In altre parole, una velocità di riga positiva muove lo sprite verso il basso, una velocità di colonna positiva muove lo sprite verso destra e viceversa.

Queste caratteristiche risultano utilissime quando si voglia ad esempio simulare, limitandosi al primo quadrante, il moto di un punto materiale soggetto a forze. L'istruzione per modificare la velocità di uno sprite è:
CALL MOTION (# numero di sprite, velocità di riga, velocità di colonna)
Il seguente programma muove a caso uno sprite, mostrando i valori relativi della velocità (fermare il programma con CLEAR):

```
200 CALL CLEAR :: RANDOMIZE
210 CALL SPRITE (# 2, 65, 2, 3, 3)
220 VR = INT (RND ★ 256) − 128 :: VC = INT (RND ★ 256) − 128
230 DISPLAY AT (2, 2) : "VELOCITA' DI RIGA :"; VR
240 DISPLAY AT (4, 2) BEEP : "VELOCITA' DI COLONNA :"; VC
250 CALL MOTION (# 2, VR, VC)
260 FOR I = 2 TO 2000 :: NEXT I
270 GOTO 220
```

Uno sprite si ferma con una istruzione del tipo:
CALL MOTION (# numero di sprite, 0, 0)
A velocità molto basse o molto alte lo sprite si muove in modo quasi perfetto. A velocità medie si nota qualche sobbalzo, ma il risultato è comunque ottimo. A velocità molto alte una sagoma perde qualcosa in definizione (è una questione ottica): in questi casi è bene scegliere per lo sprite un colore che sia ben visibile sullo sfondo.

Le informazioni sul folletto

L'Extended BASIC è dotato di una serie di istruzioni che permettono di controllare la "vita" di uno sprite. Vediamole un po' in dettaglio. L'istruzione:
CALL POSITION (# numero di sprite, pi-riga, pi-colonna)
serve a determinare la posizione di uno sprite nel preciso istante in cui viene eseguita. I valori ritornati nelle variabili numeriche "pi-riga" e "pi-colonna" sono, alla luce di quanto finora detto, numeri interi tra 1 e 256. Ad esempio l'istruzione:
CALL POSITION (# 2, RI, CO)
fa assumere alle variabili RI e CO i valori delle coordinate della posizione attuale dello sprite numero 2.
L'istruzione:
CALL COINC (# numero di sprite, pi-riga, pi-colonna, tolleranza, variabile numerica)
serve a verificare la sovrapposizione di uno sprite ad una posizione dello schermo. La posizione è quella identificata dalle variabili "pi-riga" e "pi-colonna". La "tolleranza" è il valore numerico dello scarto massimo consentito; il valore che assume la "variabile numerica" è zero se non c'è sovrapposizione, − 1 in caso

contrario. L'istruzione può anche essere utilizzata nelle forme:
a) CALL COINC (# numero di sprite, # numero di sprite, tolleranza, variabile numerica)
b) CALL COINC (ALL, variabile numerica)

Nella forma a) riporta la sovrapposizione di due sprite identificati dai rispettivi numeri. Nella forma b) riporta qualunque sovrapposizione di due o più sprite tra quelli utilizzati.
Una CALL COINC è attiva solo quando viene eseguita: nessuna sovrapposizione viene rilevata, se il computer sta eseguendo altre istruzioni. Ciò costringe a porre molta attenzione nel suo utilizzo, come vedremo in dettaglio nel seguito.

Infine l'istruzione:
CALL DISTANCE (# numero di sprite, # numero di sprite, variabile numerica)
ritorna nella "variabile numerica" il quadrato della distanza tra due sprite, calcolato applicando il ... teorema di Pitagora e riferito al pixel in alto a sinistra delle sagome. Questa istruzione può essere anche usata nella forma:
CALL DISTANCE (# numero di sprite, pi-riga, pi-colonna, variabile numerica)
in questo caso la distanza calcolata è quella fra lo sprite e la posizione dello schermo specificata dalle variabili "pi-riga" e "pi-colonna".

La morte del folletto

Uno sprite si cancella con l'istruzione:
CALL DEL SPRITE (# numero di sprite)
che può essere utilizzata anche nella forma:
CALL DEL SPRITE (ALL)
In quest'ultimo caso vengono cancellati tutti gli sprite usati in precedenza. Se il "numero di sprite" è quello di uno sprite non definito, il computer non segnala l'errore; questa caratteristica è utile in molti casi.

Note generali di utilizzo

Quanto detto finora può indurre a pensare che sia estremamente facile scrivere questo tipo di programmi. In effetti, le istruzioni a disposizione sono molte e ben "pensate", tali cioè da eliminare molti problemi. In realtà, per applicazioni in cui la velocità è essenziale le cose si complicano molto e a volte bisogna rinunciare a qualcosa per ottenere buoni risultati. Il difetto più grande è la relativa lentezza dell'interprete BASIC. Questo significa che già per velocità di grandezza intorno a 20 accade che:
- una CALL POSITION ritorna una posizione di qualche attimo indietro rispetto alla realtà;
- una sovrapposizione non viene rilevata, anche se esiste, se non per valori di tolleranza abbastanza alti;
- in caso di IF ... THEN, ogni istruzione eseguita in più fa perdere il sincronismo delle azioni.

Per questi motivi la stesura di programmi sufficientemente veloci costringe in genere a veri e propri salti mortali, come ad esempio:
- uso abbondante ed indiscriminato di GOTO, che tolgono in pratica ogni pretesa di eleganza e leggibilità al listato;
- ricorso ad espedienti per verificare le sovrapposizioni, perché una CALL COINC è praticamente inutilizzabile;
- riduzione all'osso degli effetti di contorno (ad esempio suoni, ecc.): ogni istruzione in più è un problema in più.

Il metodo probabilmente più efficace, dopo aver sviluppato l'algoritmo di movimento, è quello di affinare sul campo il programma, scegliendo alla fine i valori migliori per i parametri in gioco: velocità, posizione, tolleranze. Infatti, in pratica è impossibile sapere in anticipo, senza far girare il programma, cosa succederà sullo schermo. Il breve programma che segue è una esemplificazione degli argomenti trattati finora. Il listato è volutamente ridotto all'essenziale per permettere ad ognuno di voi di modificarlo a piacimento. Fate particolare attenzione alle tecniche utilizzate per ottenere certi risultati: con un po' di pratica vi accorgerete che tutto, entro certi limiti, può essere fatto.

Darts

Il gioco è molto semplice: avete un "lanciadardi" che si muove sulla sinistra dello schermo dall'alto in basso e spara quando premete la barra spaziatrice. Bisogna abbattere un muro di 18 mattoni. Ad ogni passaggio il muro si allontana verso destra e la velocità del lanciadardi aumenta. Un mattone, per essere distrutto, deve essere colpito due volte. Vince chi abbatte il muro nel minor numero di passaggi. Anche se relativamente poco complesso, il gioco, nell'esecuzione, è abbastanza difficile. Diciotto passaggi è già un buon risultato e sfido chiunque a farcela in meno di quattrodici. La situazione del muro è memorizzata nel vettore MU (24). Le istruzioni di movimento degli sprite sono state a lungo testate per cercare il valore migliore per i parametri in gioco. Si è verificato, infatti, che una velocità del lanciadardi maggiore di 18 fa "saltare" la verifica dei passaggi. Infine la tecnica usata per visualizzare numeri o lettere di grandezza doppia può esservi utile in altri programmi.

Vi consiglio perciò di modificare il listato secondo i vostri gusti, magari aggiungendo una sigla ed un diverso sistema per il punteggio. Vi accorgerete che non è poi così difficile costruire da soli giochi anche molto complessi: basta un po' di pratica ed un buon "rodaggio" del programma.
Buon lavoro !!!!!

Commenti al programma Darts

200 - 210 Scelta del colore dello schermo (grigio). Definizione dei caratteri grafici secondo le seguenti corrispondenze: 129 = spazio; 130 =

mezzo mattone; 131 = mattone; 132 = lanciadardi; 133 = dardo.

220 Valori iniziali del muro, rappresentato dal vettore MU (24). In particolare il valore due significa "mattone intero", uno mezzo mattone, zero nessun mattone.

230 Valori iniziali delle variabili. K è il valore delle unità e KA quello delle decine del numero di passaggi effettuati. VE è la velocità del lanciadardi, COL è la posizione del muro.

240 - 260 La variabile J controlla la fine del gioco (J = 0).
Si stampa la situazione attuale del muro.

270 Lo sprite n. 2 è il lanciadardi. Si stampa il valore delle decine del numero dei passaggi (sprite n. 4).

280 Si stampa il valore delle unità del numero dei passaggi (sprite n. 5). Si imprime velocità al lanciadardi.

290 Verifica se è stata premuta la barra spaziatrice (codice ASCII 32).

300 In caso affermativo controlla la posizione del lanciadardi: se è troppo vicino al bordo inferiore dello schermo, non spara.

310 Crea e muove il dardo verso il muro. Calcola la posizione del muro da controllare.

320 Ciclo di ritardo variabile che permette al dardo di raggiungere il muro.

330 Controlla se un mattone è stato colpito ed aggiorna la situazione del muro.

340 Cancella il dardo.

350 Verifica se il lanciadardi ha raggiunto la base dello schermo.

360 In caso affermativo calcola i nuovi valori delle variabili ed il ciclo ricomincia. Notate che il valore logico "vero" sul 99 è rappresentato da − 1. Perciò la velocità del lanciadardi sarà al massimo 18 e la posizione del muro non supererà la colonna 28.

370 - 390 Se il gioco è finito, stampa il numero dei tentativi e chiede se si vuole continuare.

400 - 410 Ciclo di attesa della risposta. Se il tasto premuto è "S" (codice ASCII 83) si ricomincia.

420 Fine programma. ■

Libri firmati JACKSON

Nicole Bréaud-Pouliquen
LA PRATICA DELL'APPLE
"Il Sistema APPLE II", il "BASIC Applesoft" il disegno e la grafica: arricchiti da esempi e esercizi.
130 pagine **L. 10.000**
Codice **341D**

F. Franceschini - F. Paterlini
Voi e il vostro Commodore 64
Uno strumento fondamentale per la comprensione e programmazione del Commodore 64. Con consigli, programmi testati, glossario e utili accenni di BASIC.
256 pagine **B L. 22.000** Codice **347**

Alan Miller
PROGRAMMI SCIENTIFICI IN PASCAL
Un'opera base per chi desidera costruirsi una "libreria" di programmi in grado di risolvere i più frequenti problemi scientifici e ingegneristici.
372 pagine **L. 25.000**
Codice **554P**

Carmine Elefante
L'home computer TI/99-4A
Il BASIC, il BASIC Esteso e il microprocessore dell'home computer della T.I. Con programmi di utilità e svago. 192 pagine **L. 15.000**
Codice **343B**

Giacomino Baisini - Giò Federico Baglioni
IL FORTH PER VIC 20 E CBM 64
La programmazione in FORTH e la sua implementazione sul Commodore VIC 20 e CBM 64.
150 pagine **L. 11.000**
Codice **527B**

Franco Filippazzi - Giulio Occhini
VOI E L'INFORMATICA
L'opera che il manager moderno non può ignorare. In 100 tavole: gli strumenti dell'Informatica, l'Informatica e l'Azienda, realtà e prospettive tecnologiche...
116 pagine **L. 15.000**
Codice **526A**

Roland Dubois
CAPIRE I MICROPROCESSORI
Un fantastico viaggio alla scoperta del "cervello" elettronico: la funzione del microprocessore, delle memorie ROM e RAM, delle interfacce...
126 pagine **L. 10.000**
Codice **342A**

Gaetano Marano
77 PROGRAMMI PER SPECTRUM
Dalla Grafica alla Business Grafica, dalla musica alle animazioni, dai giochi all'elettronica... tutte le possibilità offerte dal più piccolo dei computer.
150 pagine a colori **L. 16.000**
Codice **555A**

Rita Bonelli-Daria Gianni
ALLA SCOPERTA DEL VIC 20
Un testo chiave per imparare a conoscere e usare uno dei Personal del momento.
308 pagine **L. 22.000**
Codice **338D**
Cassetta Programmi **L. 15.000**
Floppy Programmi **L. 25.000**

nuovidea

La Biblioteca che fa testo

In busta chiusa, e senza impegno, inviate questo coupon a:
Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano
☐ Desidero ricevere gratuitamente il Catalogo Generale della Biblioteca Jackson e informazioni sulle 10 Riviste specialistiche da voi pubblicate.
(allego L. 1.000 in francobolli per contributo spese di spedizione)
☐ Desidero ricevere contrassegno il/i volume/i

_____
(pagherò al ricevimento L. .....
più L. 2.000 per contributo spese di spedizione

Nome _____ Cognome _____
Via _____
CAP _____ Città _____

TEXAS TI99/4A

Diagrammi in TI BASIC

di Sergio Borsani

Una delle più comuni rappresentazioni grafiche è data dai diagrammi a strisce o ortogrammi. Si tratta di rappresentare una serie di misure con delle strisce o delle colonne la cui lunghezza sia proporzionale alle misure stesse. Rispetto alla rappresentazione di risultati numerici sotto forma di tabella, un diagramma presenta il vantaggio di una maggiore immediatezza e consente di cogliere con un colpo d'occhio la proporzione tra i valori rappresentati o l'andamento di un fenomeno al quale essi si riferiscono. Il modo più semplice per risolvere il problema con un computer consiste nello stampare uno stesso simbolo ripetutamente in modo da formare qualcosa che assomigli ad una striscia. Alcuni computer, come il Commodore 64, posseggono già dei simboli grafici preprogrammati adatti allo scopo, con l'home computer è tuttavia semplicissimo programmare un carattere personalizzato e ripeterne la stampa in una qualsiasi posizione dello schermo. Se si desidera che le strisce risultino verticali si potrà definire il seguente carattere:

CALL CHAR (128, "7F7F7F7F7F7F7F7F")

Esso non occuperà l'intera matrice di 8 x 8 pixel ma possiede un margine a sinistra in modo che le strisce che formeranno il diagramma risultino leggermente separate una dall'altra. La stampa si ottiene poi con l'istruzione:

CALL VCHAR (R, C, 128, V)

dove R indica la riga e C la colonna, 128 indica il carattere precedentemente programmato e V il numero di volte in cui viene ripetuto lo stesso carattere, verticalmente, partendo dall'alto verso il basso.

Dimensionamento automatico

Un buon programma per la tracciatura di grafici dovrebbe possedere la caratteristica del dimensionamento automatico, cioè la striscia che rappresenta la misura maggiore dovrebbe automaticamente occupare la massima estensione consentita dal video, mentre tutte le altre risulteranno in proporzione più piccole.

Nell'esempio riportato nel listato 1 i valori variano secondo una legge esponenziale. La funzione definita alla linea 160 è la formula che si usa per calcolare l'interesse composto di un deposito in denaro. 1000 rappresenta il capitale iniziale, .14 è il tasso di interes-

```
100 REM  PROVA 1 ORTOGRAMMA
110 REM  ******************
120 CALL CLEAR
130 CALL CHAR(128,"7F7F7F7F7F7F7F7F")
140 CALL CHAR(136,"FF")
150 CALL COLOR(13,5,1)
160 DEF F(T)=1000*(1+.14)^T
170 MAX=F(28)
180 U=MAX/20
190 CALL HCHAR(22,3,136,28)
200 FOR J=1 TO 28
210 LUNG=INT(F(J)/U)
220 IF LUNG=0 THEN 240
230 CALL VCHAR(22-LUNG,J+2,128,LUNG)
240 NEXT J
250 PRINT TAB(7);"<BATTI UN TASTO>"
260 CALL KEY(0,K,S)
270 IF S=0 THEN 260
280 CALL CLEAR
290 END
```

Listato 1. *Programma "Prova 1 Ortogramma".*

se (14%) e T è il numero degli anni. Poiché il programma è fatto per stampare un grafico con 28 colonne è chiaro che in questo caso particolare il valore massimo da rappresentare si trova in corrispondenza di T = 28, cioè F (28).

È per questa ragione che alla linea 170 questo valore è stato preso come massimo. In altri casi si dovrà seguire una via diversa e andrà benissimo una piccola routine come quella riportata dal manuale a pagina 60. Chiamato MAX il valore massimo, se si desidera che la colonna più lunga non superi la lunghezza di 20 caratteri, sarà U = MAX/20 il valore unitario, cioè il valore di ogni tratto elementare che forma la striscia. La stampa dell'intero diagramma è compiuta dal ciclo FOR NEXT alle linee 200-240. LUNG è la lunghezza o il numero di caratteri di ogni striscia e l'istruzione 230 è fatta in modo che tutte le colonne abbiano la base posta in corrispondenza della riga 21 dello schermo. Eseguendo il programma del listato 1 si noterà che l'aspetto del diagramma non è molto soddisfacente. Il problema è che la lunghezza delle strisce varia per valori discreti abbastanza elevati e che misure differenti possono essere rappresentate con colonnine formate dallo stesso numero di caratteri. Il diagramma potrebbe essere migliorato se si potesse disporre di caratteri che rappresentano una parte frazionaria di quello fino ad ora usato.

Definiamo altri simboli grafici

Si osservi il listato 2. Oltre al carattere 128 ne troviamo altri sette (il carattere 136 serve solo a

```
100 REM  PROVA 2 ORTOGRAMMA
110 REM  ******************
120 CALL CLEAR
130 CALL CHAR(128,"7F7F7F7F7F7F7F7F")
140 CALL CHAR(129,"00000000000007F")
150 CALL CHAR(130,"0000000000007F7F")
160 CALL CHAR(131,"000000000007F7F7F")
170 CALL CHAR(132,"00000007F7F7F7F")
180 CALL CHAR(133,"0000007F7F7F7F7F")
190 CALL CHAR(134,"00007F7F7F7F7F7F")
200 CALL CHAR(135,"007F7F7F7F7F7F7F")
210 CALL CHAR(136,"FF")
220 CALL COLOR(13,5,1)
230 DEF F(T)=1000*(1+.14)^T
240 MAX=F(28)
250 U=MAX/20
260 CALL HCHAR(22,3,136,28)
270 FOR J=1 TO 28
280 LUNG=INT(F(J)/U)
290 IF LUNG=0 THEN 310
300 CALL VCHAR(22-LUNG,J+2,128,LUNG)
310 DIFF=F(J)/U-LUNG
320 PF=INT(8*DIFF)
330 IF PF=0 THEN 350
340 CALL HCHAR(21-LUNG,J+2,128+PF)
350 NEXT J
360 PRINT TAB(7);"<BATTI UN TASTO>"
370 CALL KEY(0,K,S)
380 IF S=0 THEN 370
390 CALL CLEAR
400 END
```

Listato 2. *Programma "Prova 2 Ortogramma".*

tracciare la base del diagramma). Ogni simbolo grafico occupa una matrice di 8 x 8 punti video (pixel), ebbene, il primo è alto 8 pixel e rappresenta il carattere unitario, gli altri sono alti rispettivamente 1, 2, 3, ..., 7 pixel. Ora che si posseggono i caratteri adatti, il nostro compito sarà quello di stampare per ogni striscia un certo numero di caratteri "unitari", come è stato fatto in precedenza, e, sopra questi, collocare il carattere "frazionario" più adatto. Senza riscrivere l'intero programma, è sufficiente aggiungere le linee dove vengono definiti i nuovi caratteri (vedi listato 2). Il programma procede poi allo stesso modo fino alla linea 300. Alla linea successiva si calcola la differenza (DIFF) tra il valore che si voleva rappresentare e quello che effettivamente si è rappresentato o meglio tra la lunghezza della colonna quale doveva essere e quale effettivamente è. Tale differenza è sempre inferiore all'unità ed è espressa in decimi. Tuttavia i simboli grafici definiti variano tra di loro di uno o più pixel ed un pixel rappresenta un ottavo di carattere, non un decimo. La linea 320 del programma trasforma la parte decimale in un numero compreso tra 0 e 7. Un esempio numerico chiarirà meglio questo punto. Se la differenza DIFF è uguale a 0.4, sarà PF = INT (8 ★ 0.4), cioè PF = 3 e alla linea 340 del programma si stampa il carattere 128 + PF, cioè il 131 che è alto i 3/8 del carattere unitario ed è quello che meglio rappresenta i 4 decimi mancanti alla colonnina. I valori utilizzati da questo programma sono gli stessi usati in precedenza ma la rappresentazione grafica è ora notevolmente più precisa.

COMMODORE 64

Il 64 e la memoria nascosta

di Alessandro Guida

È noto a tutti che il 64 è pubblicizzato dalla Commodore come un sistema da 64 Kbyte di RAM. Al contrario, all'accensione sullo schermo appare un deludente "38911 BYTES FREE".
Ma niente paura i restanti 24 Kbyte di RAM ci sono, anche se nascosti.

I banchi di memoria

Il microprocessore di cui è dotato il 64, il 6510, ha un bus indirizzi da 16 bit. Questo significa che è in grado di indirizzare 65536 locazioni, cioè 64 Kbyte di memoria. Di questi 64 Kbyte però, 16 sono riservati alla ROM contenente l'interprete BASIC e il sistema

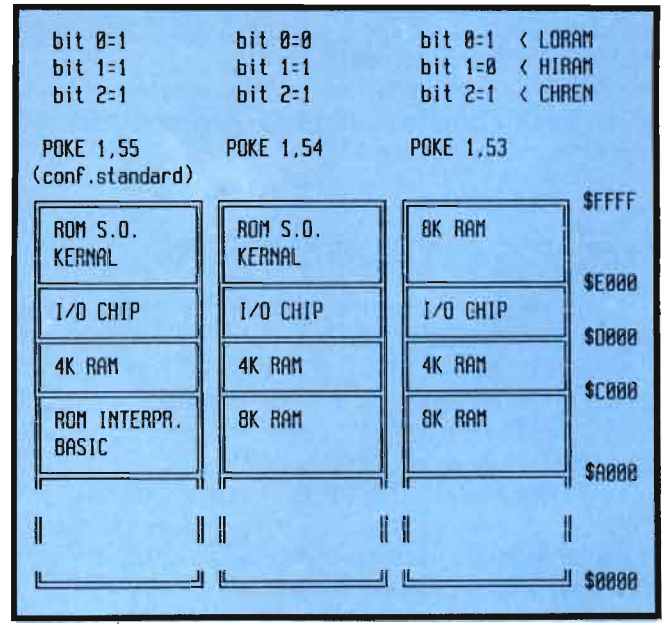


Figura 1. *Le tre più importanti configurazioni della memoria del C 64 con il relativo POKE nel registro 1.*

PERSONAL SOFTWARE

Stid Poker

"Come giocare a Poker contro il Texas TI99 e rischiare di perdere"

di *Giuseppe Persiano*

Lo Stid Poker ricalca, fatte poche eccezioni, un giochino derivato dal Poker chiamato Telesa secca, che si gioca in quattro, con le carte dal sette all'asso. Le regole sono semplici: innanzitutto viene data a ciascuno una carta coperta, che rimane tale durante tutto il gioco, in seguito quattro scoperte. Si può scommettere o rilanciare dopo la distribuzione di ciascuna carta scoperta (non dopo la prima).
Dopo aver distribuito l'ultima carta (e rilanciato) si scopre la carta coperta di ciascun giocatore ancora in gioco per proclamare ii vincitore.
Il meccanismo dei punti è analogo al Poker normale.
Lo Stid Poker è diverso dalla Telesa perché il primo a parlare, in ogni turno, è chi ha effettuato l'ultimo rilancio da quello precedente o, se ci si trova al primo turno (cioè dopo la seconda carta), è il primo di mano. Una seconda differenza riguarda il punteggio: in caso di parità vince chi ha effettuato l'ultimo rilancio.

Dopo questa nota di carattere introduttivo vediamo come giocare a Stid Poker. Dopo aver dato il RUN sullo schermo appare la scritta "QUANTI SIETE? (0-4)", occorre quindi digitare ii numero dei partecipanti, automaticamente eventuali giocatori mancanti saranno guidati dal TI. Se ii numero dato è diverso da zero seguirà la richiesta dei nomi dei giocatori che verranno stampati sopra ii capitale individuale.
A questo punto vengono distribuite le prime due carte e appare la scritta "STO PENSANDO". Scomparsa questa, si può dar via alle giocate e ad eventuali rilanci.
Il giocatore indicato dall'asterisco può:
1) guardare la propria carta coperta premendo il tasto H;
2) ricoprire la prima carta dopo averla guardata premendo il tasto S;
3) affidare la decisione al computer premendo ii tasto A;
4) rinunciare al gioco, cioè passare, premendo il tasto P;
5) vedere il rilancio di un altro giocatore premendo ii tasto V;
6) battere o "fare cip" premendo il tasto V;
7) rilanciare premendo ii tasto R e digitare l'importo del rilancio usando i tasti numerici. Nel caso in cui ci si sbagli basta premere ii tasto "." (punto) per annullare la cifra introdotta.
Naturalmente ii piatto stampato in alto e al centro dello schermo e la dote di ogni giocatore vengono aggiornati automaticamente.
Se rlmane in gioco un solo giocatore questo è ovviamente il vincitore, altrimenti il computer provvede a indicare il vincitore sullo schermo.

Matrici usate

SEGNO (i) - Indica il codice del simbolo i-esimo. Si sottolinea che il sette è il primo carattere, l'asso l'ottavo. Per il dieci è stato adottato il carattere D.
Z (i) - Indica il codice dell'-iesimo seme. Si precisa che cuori è il primo seme, quadri ii secondo e cosi via.
COLORE (i, j)- Indica quante carte del j-esimo seme possiede il giocato-

Listato 1. *Il programma BASIC.*

```
1 OPTION BASE 1
2 DIM RIGA(4),COLONNA(4),DOS(4,5
),PASSO(4),SEGNO(8),CU(4,8),V(4)
,COM(4),SEMI(4),B$(4),P1(4),C(4,
5),S(4,5),Z(4)
3 DIM VERSA(4),DOTE(4),COLORE(4,
5),ROS(4,8),CARTA(8)
4 CALL CLEAR
5 FOR I=1 TO 4
6 READ RIGA(I),COLONNA(I)
7 FOR B=1 TO 5
8 READ DOS(I,B)
9 NEXT B
10 NEXT I
11 CALL CHAR(105,"DCDCDCDCDCDCDC
DC")
12 CALL COLOR(15,7,16)
13 CALL COLOR(16,2,16)
14 CALL SCREEN(5)
15 CALL CHAR(97,"00")
16 READ SEGNO(1)
17 FOR T=2 TO 8
18 READ SEGNO(T)
19 CALL COLOR(T,2,16)
20 NEXT T
21 CALL COLOR(9,7,16)
22 CALL CHAR(144,"0066FFFFFF7E3C
18")
23 CALL CHAR(145,"183C7EFFFF7E3C
18")
```

Seguito listato 1.

```
24 CALL CHAR(153,"183C7EFFFF5A18
18")
25 CALL CHAR(152,"183C185AFF5A18
18")
26 READ Z(1),Z(2),Z(3),Z(4),A$,D
$
27 FOR I=1 TO 20
28 CALL HCHAR(12,5+I,ASC(SEG$(A$
,I,1)))
29 NEXT I
30 CALL KEY(5,K,N)
31 IF N=0 THEN 30
32 IF K=13 THEN 37
33 IF (K<48)+(K>52)THEN 30
34 CALL HCHAR(12,26,K)
35 K1=K
36 GOTO 30
37 FOR I=K1-47 TO 4
38 COM(I)=1
39 NEXT I
41 FOR Y=1 TO K1-48
42 PRINT "NOME";Y,
43 INPUT B$(Y)
44 NEXT Y
45 DEF INC4(B)=-(B+1*(B+1<>5)-(
B+1=5)
46 FOR W=1 TO 4
47 DOTE(W)=1000
48 GOSUB 17000
49 GOSUB 20000
50 NEXT W
140 CALL HCHAR(17,4,32,285)
141 CALL HCHAR(5,4,32,285)
142 PIATTO=0
143 GOSUB 19000
144 FOR I=1 TO 4
145 FOR B=1 TO 8
146 CU(I,B)=1
147 NEXT B
148 V(I)=1
149 SEMI(I)=0
150 NEXT I
151 FOR A=1 TO 5
152 FOR F=1 TO 4
153 C(F,A)=0
154 S(F,A)=0
```

Seguito listato 1.

```
155 COLORE(F,A)=0
156 NEXT F
157 NEXT A
158 FOR G=1 TO 8
159 FOR T=1 TO 4
160 ROS(T,G)=0
161 NEXT T
162 CARTA(G)=0
163 NEXT G
164 M=0
165 ULT=1
166 ANDATI=0
167 REM  FINE INIZIALIZZAZIONE V
ARIABILI
994 REM  INIZIO PROGRAMMA PRINCI
PALE
995 M=M+1
996 CALL HCHAR(5,1,32,64)
997 CALL HCHAR(17,1,32,64)
1000 FOR GIOC=1 TO 4
1002 PASSO(GIOC)=0
1005 IF V(GIOC)=0 THEN 1071
1010 RANDOMIZE
1015 C(GIOC,M)=INT(RND*8)+1
1020 RANDOMIZE
1025 S(GIOC,M)=INT(RND*4)+1
1030 IF CU(S(GIOC,M),C(GIOC,M))=
0 THEN 1010
1035 CU(S(GIOC,M),C(GIOC,M))=0
1040 IF M=1 THEN 1060
1045 REM  AGGIORNAMENTO VARIABIL
I
1050 SEMI(S(GIOC,M))=SEMI(S(GIOC
,M))+1
1055 CARTA(C(GIOC,M))=CARTA(C(GI
OC,M))+1
1060 ROS(GIOC,C(GIOC,M))=ROS(GIO
C,C(GIOC,M))+1
1065 COLORE(GIOC,S(GIOC,M))=COLO
RE(GIOC,S(GIOC,M))+1
1070 GOSUB 13000
1071 NEXT GIOC
1072 IF M=1 THEN 995
1073 FOR Q=1 TO 12
1074 CALL HCHAR(14,8+Q,ASC(SEG$(
D$,Q,1)))
```

Seguito listato 1.

```
1075 NEXT Q
1076 FOR GIOC=1 TO 4
1077 IF V(GIOC)=0 THEN 1080
1078 ON M-1 GOSUB 23000,23100,23
100,23300
1080 NEXT GIOC
1081 CALL HCHAR(14,9,32,12)
1082 W=ULT
1083 CALL VCHAR(RIGA(W)+2,DOS(W,
1)+3,42,4)
1084 ON COM(W)+1 GOSUB 25000,260
00
1085 CALL VCHAR(RIGA(W)+2,DOS(W,
1)+3,32,4)
1086 W=INC4(W)
1096 IF ANDATI<3 THEN 1110
1101 VIN=ULT
1104 GOSUB 28000
1107 GOTO 140
1110 IF V(W)=0 THEN 1086
1111 IF W<>ULT THEN 1083
1113 IF M<5 THEN 995
1116 FOR F=1 TO 4
1119 IF V(F)=0 THEN 1125
1122 GOSUB 14999
1123 P1(F)=10*MAX+DOR
1125 NEXT F
1128 GOSUB 27000
1131 GOSUB 28000
1134 GOTO 140
1137 REM  *FINE PROGRAMMA PRINCI
PALE*
12997 REM ==================

12998 REM  ROUTINE GRAFICA
12999 REM ==================

13000 IF M=1 THEN 13035
13005 FOR R=DOS(GIOC,M)TO DOS(GI
OC,M)+2
13010 CALL VCHAR(RIGA(GIOC)+2,R,
97,4)
13015 NEXT R
13020 CALL HCHAR(RIGA(GIOC)+2,DO
S(GIOC,M),SEGNO(C(GIOC,M)))
13025 CALL HCHAR(RIGA(GIOC)+5,DO
S(GIOC,M),Z(S(GIOC,M)))
13030 RETURN
13035 FOR R=DOS(GIOC,1)TO DOS(GI
OC,1)+2
13040 CALL VCHAR(RIGA(GIOC)+2,R,
```

Seguito listato 1.

```
105,4)
13045 NEXT R
13050 RETURN
14947 REM  =================
14948 REM  ROUTINE RICONOSCIMENT
O PUNTI
14949 REM ===================

14950 ROS(F,C(F,1))=ROS(F,C(F,1)
)-1
14955 P=1
14960 GOSUB 14999
14965 P=0
14975 ROS(F,C(F,1))=ROS(F,C(F,1)
)+1
14980 RETURN
14999 MAX=0
15000 FOR HC=1 TO 8
15001 IF ROS(F,HC)>=2 THEN 15004

15002 NEXT HC
15003 ON P+1 GOTO 15028,15014
15004 MAX=1
15005 DOR=HC
15006 FOR HD=HC+1 TO 8
15007 IF ROS(F,HD)>=2 THEN 15010

15008 NEXT HD
15009 GOTO 15021
15010 MAX=2
15011 DOR=HD
15012 IF ROS(F,HC)=3 THEN 15018

15013 IF ROS(F,HD)=3 THEN 15015

15014 RETURN
15015 DOR=HD
15016 MAX=5
15017 RETURN
15018 DOR=HC
15019 MAX=5
15020 RETURN
15021 IF ROS(F,HC)>=3 THEN 15023

15022 RETURN
15023 MAX=3
15024 IF ROS(F,HC)=4 THEN 15026

15025 RETURN
15026 MAX=6
15027 RETURN
```

Seguito listato 1.

```
15028 FOR HS=1 TO 4
15029 IF (ROS(F,HS)=1)*(ROS(F,HS
+1)=1)*(ROS(F,HS+2)=1)*(ROS(F,HS
+3)=1)*(ROS(F,HS+4)=1)THEN 15032

15030 NEXT HS
15031 GOTO 15034
15032 MAX=4
15033 DOR=HSC
15034 FOR HCL=1 TO 4
15035 IF COLORE(F,HCL)=5 THEN 15
038
15036 NEXT HCL
15037 IF MAX=0 THEN 15044 ELSE 1
5039
15038 MAX=MAX+6
15039 RETURN
15044 FOR Y=1 TO 5
15046 DOR=-DOR*(C(F,Y)<DOR)-C(F,
Y)*(C(F,Y)>DOR)
15048 NEXT Y
15050 RETURN
16997 REM ==================
16998 REM  ROUTINE DI STAMPA DOT
I
16999 REM ==================
17000 A$=STR$(DOTE(W))
17005 IF LEN(A$)>=4 THEN 17020
17010 A$="0"&A$
17015 GOTO 17005
17020 FOR I=1 TO LEN(A$)
17025 CALL HCHAR(RIGA(W)-3,COLON
NA(W)+I,ASC(SEG$(A$,I,1)))
17030 NEXT I
17035 RETURN
18997 REM  =================
18998 REM  ROUTINE STAMPA PIATTO

18999 REM  =================
19000 A$=STR$(PIATTO)
19005 IF LEN(A$)>=4 THEN 19020
19010 A$="0"&A$
19015 GOTO 19005
19020 FOR I=1 TO LEN(A$)
19025 CALL HCHAR(1,12+I,ASC(SEG$
(A$,I,1)))
19030 NEXT I
19035 RETURN
19997 REM  =================
19998 REM  ROUTINE STAMPA NOMI
19999 REM  =================
```

Seguito listato 1.

```
20000 FOR D=1 TO LEN(B$(W))
20005 CALL HCHAR(RIGA(W)-4,COLON
NA(W)+D,ASC(SEG$(B$(W),D,1)))
20010 NEXT D
20015 RETURN
22999 REM  =================
23000 REM  CALCOLO DI PASSO(GIOC
) PER M=2
23001 REM  =================
23002 FOR CA=1 TO 8
23004 IF ROS(GIOC,CA)=0 THEN 230
08
23006 PASSO(GIOC)=PASSO(GIOC)+(R
OS(GIOC,CA)-0.5)^2*CA/4+(4-CARTA
(CA))^2-25*(ROS(GIOC,CA)=2)
23008 NEXT CA
23010 RETURN
23097 REM  ==================

23098 REM   CALCOLO DI PASSO(GIO
C)  M=3   M=4
23099 REM  ==================

23100 FOR F=1 TO 4
23101 IF V(F)=0 THEN 23104
23102 ON -(F=GIOC)+1 GOSUB 14950
,14999
23103 Pl(F)=10*MAX+DOR
23104 NEXT F
23105 IF M=4 THEN 23112
23106 FOR H=1 TO 8
23108 IF ROS(GIOC,H)=0 THEN 2311
1
23110 PASSO(GIOC)=PASSO(GIOC)+((
ROS(GIOC,H)-1)^2*H/4+(4-CARTA(H)
/3)^2)/9
23111 NEXT H
23112 PASSO(GIOC)=PASSO(GIOC)+7*
(Pl(1)>Pl(GIOC))+7*(Pl(2)>Pl(GIO
C))+7*(Pl(3)>Pl(GIOC))+Pl(GIOC)*
2
23114 PASSO(GIOC)=PASSO(GIOC)+7*
(Pl(4)>Pl(GIOC))
23116 RETURN
23297 REM   ================

23298 REM  CALCOLO DI PASSO(GIOC
)  M=5
23299 REM  ==================

23300 FOR F=1 TO 4
```

Seguito listato 1.

```
23302 IF V(F)=0 THEN 23306
23304 ON (F=GIOC)+2 GOSUB 14999,
14950
23305 P1(F)=10*MAX+DOR
23306 NEXT F
23308 PASSO(GIOC)=PASSO(GIOC)+10
0*(P1(1)>P1(GIOC))+100*(P1(2)>P1
(GIOC))+100*(P1(3)>P1(GIOC))
23310 PASSO(GIOC)=PASSO(GIOC)+10
0*(P1(4)>P1(GIOC))+3*P1(GIOC)
23312 RETURN
24997 REM  ================
24998 REM  ROUTINE DI DECISIONE

24999 REM  ================

25000 CALL KEY(5,K,N)
25002 IF N=0 THEN 25000
25004 IF (K<>86)*(K<>82)*(K<>80)
*(K<>65)*(K<>72)*(K<>83)THEN 250
00
25006 ON -1*(K=72)-2*(K=83)-3*(K
=65)-4*(K=80)-5*(K=82)-6*(K=86)G
OTO 25008,25022,25032,25064,2503
7,25068
25008 FOR R=DOS(W,1)TO DOS(W,1)+
2
25010 CALL VCHAR(RIGA(W)+2,R,97,
4)
25012 NEXT R
25014 CALL VCHAR(RIGA(W)+2,DOS(W
,1),SEGNO(C(W,1)))
25016 CALL HCHAR(RIGA(W)+5,DOS(W
,1),Z(S(W,1)))
25018 GOTO 25000
25022 FOR R=DOS(W,1)TO DOS(W,1)+
2
25024 CALL VCHAR(RIGA(W)+2,R,105
,4)
25026 NEXT R
25028 GOTO 25000
25032 ON M-1 GOSUB 26050,26100,2
6300,26500
25034 RETURN
25037 CALL HCHAR(RIGA(W)-2,COLON
NA(W),32,5)
25038 A$=""
25040 Z1=0
25041 CALL KEY(5,K,N)
25042 IF N=0 THEN 25041
25043 IF K=46 THEN 25037
```

Seguito listato 1.

```
25045 IF K=13 THEN 25058
25046 IF (K<48)+(K>57)THEN 25041

25050 CALL HCHAR(RIGA(W)-2,COLON
NA(W)+Z1,K)
25052 A$=A$&CHR$(K)
25054 Z1=Z1+1
25056 GOTO 25041
25058 RIL=VAL(A$)
25060 GOSUB 30200
25062 RETURN
25064 GOSUB 30000
25066 RETURN
25068 GOSUB 30100
25070 RETURN
25097 REM  ================
25098 REM  ROUTINE DI DECISIONE
AUTOMATICA
25099 REM  ================
26000 ON M-1 GOSUB 26050,26100,2
6300,26500
26005 RETURN
26050 IF PASSO(W)>=13 THEN 26059

26053 GOSUB 30000
26056 RETURN
26059 IF PASSO(W)>=30 THEN 26068

26063 ON -(VERSA(W)<=PASSO(W))-2
*(VERSA(W)>PASSO(W))GOSUB 30100
30000
26065 RETURN
26068 A=RND*100
26071 IF A>=65 THEN 26080
26074 GOSUB 30100
26077 RETURN
26080 RIL=INT(PASSO(W)/8)+2
26083 GOSUB 30200
26089 RETURN
26100 IF PASSO(W)<35 THEN 26108

26102 RIL=INT(PASSO(W)/16)+2
26104 GOSUB 30200
26106 RETURN
26108 IF PASSO(W)>20 THEN 26114

26110 GOSUB 30000
26112 RETURN
26114 ON -(VERSA(W)>PASSO(W)/4)-
2*(VERSA(W)<=PASSO(W)/4)GOSUB 30
000,30100
```

Stid Poker

Seguito listato 1.

```
26116 RETURN
26300 ON -(PASSO(W)<0)+2*((PASSO
(W)>=0)*(PASSO(W)<30))-3*(PASSO(
W)>=30)GOTO 26302,26306,26309
26302 GOSUB 30000
26304 RETURN
26306 ON -(VERSA(W)<PASSO(W)/5)-
2*(PASSO(W)<=5*VERSA(W))GOSUB 30
100,30000
26308 RETURN
26309 RIL=INT(PASSO(W)/20)+1
26310 ON -(VERSA(W)>PASSO(W)/3)-
2*(VERSA(W)<=PASSO(W)/3)GOSUB 30
100,30200
26312 RETURN
26500 IF PASSO(W)>=33 THEN 26506

26502 GOSUB 30000
26504 RETURN
26506 IF VERSA(W)<PASSO(W)/10 TH
EN 26514
26508 GOSUB 30100
26510 RETURN
26514 RIL=INT(PASSO(W)/15)
26516 ON -(PASSO(W)>50)-2*(PASSO
(W)<=50)GOSUB 30200,30100
26518 RETURN
26999 REM  ROUTINE RICONOSCIMENT
O VINCITORE
27000 FOR T=1 TO 4
27003 IF V(T)=0 THEN 27024
27006 FOR B=1 TO 4
27009 IF V(B)=0 THEN 27015
27012 IF P1(T)<P1(B)THEN 27024
27015 NEXT B
27018 VIN=T
27021 RETURN
27024 NEXT T
27027 VIN=ULT
```

Seguito listato 1.

```
27030 RETURN
28000 A$="VINCE"
28001 DOTE(VIN)=DOTE(VIN)+PIATTO

28002 CALL HCHAR(RIGA(VIN)-1,COL
ONNA(VIN),42,13)
28004 FOR I=1 TO 5
28006 CALL HCHAR(RIGA(VIN)-1,COL
ONNA(VIN)+3+I,ASC(SEG$(A$,I,1)))

28008 NEXT I
28010 IF ANDATI=3 THEN 28028
28014 FOR W=1 TO 4
28016 IF V(W)=0 THEN 28027
28018 FOR R=DOS(W,1)TO DOS(W,1)+
2
28020 CALL VCHAR(RIGA(W)+2,R,97,
4)
28022 NEXT R
28024 CALL HCHAR(RIGA(W)+5,DOS(W
,1),Z(S(W,1)))
28026 CALL HCHAR(RIGA(W)+2,DOS(W
,1),SEGNO(C(W,1)))
28027 NEXT W
28028 W=VIN
28030 GOSUB 17000
28032 CALL KEY(5,K,N)
28034 IF N=0 THEN 28032
28036 RETURN
29997 REM  ================
29998 REM  ROUTINE DI PASSO
29999 REM  ================
30000 IF VERSA(W)=0 THEN 30104
30002 VERSA(W)=0
30003 V(W)=0
30004 A$="PASSO"
30005 ANDATI=ANDATI+1
30006 CALL HCHAR(RIGA(W)-1,COLON
NA(W),32,9)
```

re i.

ROS (i, j) - Indica quante carte col j-esimo simbolo possiede il giocatore i.

C (i, j) - Indica il simbolo riportato sulla j-esima carta del giocatore i.

S (i, j) - Indica il seme della j-esima del giocatore i.

SEMI (i) - Indica quante carte col seme i sono già uscite, escludendo le carte coperte.

CARTA (i) - Riporta il numero di carte col simbolo i-esimo già uscite, escludendo le carte coperte.

V (i) - Se V (i) = 1 allora il giocatore i-esimo è ancora in gioco altrimenti ha rinunciato al gioco.

CU (i, j) - Se CU (i, j) = 1 allora la carta di seme i e simbolo j è già stata distribuita.

COM (i) - Se COM (i) = 1 allora il giocatore i è guidato da TI; B$ è la matrice che contiene i nomi.

P1 - Contiene i codici dei punti di ciascun giocatore; più è alto il valore nel corrispondente indice, più è alto il punto del giocatore.

Le matrici RIGA, COLONNA, DOS contengono dei valori di riferimento per la grafica.

La matrice PASSO contiene un valore per ciascun contendente che indica le possibilità di vincita di ciascun giocatore.

VERSA, DOTE sono matrici che indicano rispettivamente quanto de-

Stid Poker

Seguito listato 1.

```
30008 FOR AD=0 TO LEN(A$)-1
30010 CALL HCHAR(RIGA(W)-1,COLON
NA(W)+AD,ASC(SEG$(A$,AD+1,1)))
30012 NEXT AD
30014 RETURN
30097 REM  ================
30098 REM  ROUTINE DI VEDO
30099 REM  ================
30100 DOTE(W)=DOTE(W)-VERSA(W)
30101 IF VERSA(W)=0 THEN 30104
30102 A$="O.K."
30103 GOTO 30105
30104 A$="CIP"
30105 PIATTO=PIATTO+VERSA(W)
30106 GOSUB 30006
30107 GOSUB 17000
30108 GOSUB 19000
30109 VERSA(W)=0
30110 RETURN
30197 REM  ===============
30198 REM  ROUTINE DI RILANCIO
30199 REM  ===============
```

Seguito listato 1.

```
30200 A$="RILANCIO"
30201 ULT=W
30202 GOSUB 30006
30203 PASSO(W)=PASSO(W)-2*M-2
30204 A$=STR$(RIL)
30205 CALL HCHAR(RIGA(W)-2,COLON
NA(W),32,8)
30206 FOR AD=1 TO LEN(A$)
30208 CALL HCHAR(RIGA(W)-2,COLON
NA(W)+AD,ASC(SEG$(A$,AD,1)))
30210 NEXT AD
30212 FOR AC=1 TO 4
30214 VERSA(AC)=V(AC)*(VERSA(AC)
+RIL)
30216 NEXT AC
30217 PIATTO=PIATTO+VERSA(W)
30218 DOTE(W)=DOTE(W)-VERSA(W)
30220 VERSA(W)=0
30221 GOSUB 19000
30222 GOSUB 17000
30224 RETURN
31000 DATA 7,4,3,8,10,12,14,7,20
,19,24,26,28,30,19,4,3,8,10,12,1
4,19,20
31010 DATA 19,24,26,28,30,55,56,
57,68,74,81,75,65,144,145,153,15
2,"QUANTI SIETE? (0-4) ","STO PE
NSANDO"
```

ve pagare un giocatore per giocare e quanto gli è rimasto.

Commento al listato

1-24 Dimensionamento delle matrici, definizione dei caratteri grafici, inizializzazione.
27-44 Richiesta del numero dei giocatori. La linea 33 è stata inserita per evitare l'inserimento di valori errati. Richiesta del nome dei giocatori.
45 Definisce l'addizione del modulo 4.
46-167 Azzera i contenuti delle matrici per iniziare una nuova partita.
996-997 Cancella la parte di schermo in cui erano riportati i rilanci del precedente turno.
1000-1071 Distribuisce la carta e aggiorna i contenuti delle matrici SEMI, CARTA, COLORE, ROS.
1073-1134 Dà la parola ai giocatori ancora in gioco, chiama la routine di decisione automatica, verifica che non siano in gioco meno di due giocatori altrimenti proclama vincitore l'ultimo giocatore rimasto.
13000 Routine grafica; visualizza le carte man mano che vengono generate; se si è alla prima carta ne stampa il dorso.
14950 Questa routine viene usata per riconoscere i punti dei giocatori; se chiamata dalla riga 14950 nasconde la prima carta; tale evenienza si presenta quando la routine viene chiamata dalla routine decisionale di un altro giocatore per vedere il punto esibito dalle carte scoperte.
17000-19000 Visualizzano la dote, il nome di ciascun giocatore nonché l'ammontare del piatto.
23000-23312 Queste routine servono per il calcolo dei valori della matrice PASSO; per tali calcoli si tiene conto delle proprie carte, di quelle degli altri giocatori (escludendo quelle coperte), dei punti esibiti dagli altri giocatori (a tal fine viene chiamata la routine 14950).
25000 Mediante una CALL KEY viene interrogata la tastiera per conoscere la decisione dei giocatori.
26000-26520 Queste routine servendosi dei valori della matrice PASSO, decidono la strategia di gioco di ciascun giocatore guidato dal TI; oltre che della matrice PASSO viene tenuto conto anche della matrice VERSA che indica quanto deve versare ciascun giocatore per restare in gioco.
30000 Queste ultime routine vengono chiamate dalle routine decisionali per rendere noto il comportamento di ciascun giocatore (cioè se rilancia, se vede oppure passa) e aggiornano i valori del piatto, dalle matrici VERSA, V, DOTE a seconda degli sviluppi del gioco.
Come appare evidente, s'é fatto largo uso dei valori attribuiti ad ogni espressione relazionale dal computer, a secondo se questa è vera oppure falsa; in tal modo s'è potuto simulare l'istruzione di salto condizionato nelle subroutine condizionato. Non tutti sappiamo che il TI BASIC non accetta una linea del genere: 100 IF A = 4 THEN GOSUB 200 ELSE GOSUB 300; ma possiamo scrivere: 100 ON − (A = 4) − 2 ★ (A 4) GOSUB 200, 300;
in più, il confronto si può estendere anche a più di una condizione, come fatto alla linea 26300. ■

D = INT n/256
E = n − 256 x D

dove n è l'indirizzo desiderato.
Si noti che il valore di DE (figura 1) deve essere uguale al valore di HL (figura 2).
Vengono indicati anche un programma (listato 1) per il caricamento della routine (figura 1 + figura 2) ed un programma dimostrativo (listato 3).

TEXAS TI 99/4A

Apertura di un file nel modo APPEND

di Sergio Borsani

Uno dei problemi che affliggono l'utilizzatore di un sistema di memoria a cassette è costituito dalle limitazioni nell'uso dei file. Sono infatti consentiti soltanto file sequenziali e se si desidera simulare un accesso diretto ai record bisogna caricare nella memoria centrale l'intero file, che ovviamente dovrà avere un'ampiezza proporzionata ai Kbyte disponibili.
Analoghe considerazioni si possono fare quando si deve modificare un file sequenziale o quando si deve ampliarlo, aggiungendo nuovi record a quelli già esistenti.
Se si dispone di due registratori, uno potrà essere utilizzato per leggere il vecchio file e l'altro per scrivere il file aggiornato.
Disponendo di un solo registratore si caricherà in memoria l'intero file, si apporteranno ad esso le modifiche e si passerà quindi alla sua registrazione.
C'è un'eccezione a questa sequenza di operazioni qualora lo scopo sia semplicemente l'aggiunta di nuovi record.
Le caratteristiche di apertura di un file con l'uso di un registratore sono riportate nel manuale, sotto la voce OPEN, a pag. 120.
Da esse si desume che gli unici modi consentiti sono quelli in INPUT ed in OUTPUT; gli altri, UPDATE e APPEND, sono disponibili solo con il sistema a dischi.

In realtà, il modo APPEND può essere ottenuto con un breve programma scritto in TI BASIC; in altre parole è possibile aggiungere nuovi record alla fine di un file senza che si verifichino inconvenienti in fase di lettura. Il trucco, se vogliamo chiamarlo così, è abbastanza banale. Si tratta di aprire il file in input e di leggere i record preesistenti conoscendo esattamente il loro numero, senza tuttavia mantenerli in memoria, quindi si chiude il file e si riapre in output per la registrazione di ulteriori record. Una variabile dovrà fungere da contatore per tenere aggiornato il numero totale dei record scritti.
Tale numero andrà segnato sull'etichetta della cassetta, assieme a quello indicato dal tape counter che segna l'inizio del file.

Listato 1. *Il programma BASIC.*

```
100 REM  PROVA FILE
110 REM  **********
120 CALL CLEAR
130 OPEN #1:"CS1",SEQUENTIAL,OUTPUT,INTE
RNAL,FIXED
140 REC1$="RECORD N.1"
150 PRINT #1:REC1$
160 CLOSE #1
170 PRINT
180 PRINT "NUMERO TOTALE DEI RECORD = 1"
190 PRINT
200 PRINT "VUOI SCRIVERE ALTRI RECORD?"
210 INPUT "(Y/N) ":R$
220 IF R$="Y" THEN 260
230 IF R$="N" THEN 250
240 GOTO 210
250 END
260 PRINT
270 PRINT "PRIMA LEGGIAMO I RECORD"
280 PRINT "PRECEDENTI."
290 INPUT "QUANTI NE DEVO LEGGERE? ":N
300 TOT=N
310 OPEN #1:"CS1",SEQUENTIAL,INPUT ,INTE
RNAL,FIXED
320 PRINT
330 PRINT "RECORD PRECEDENTI"
340 FOR J=1 TO N
350 INPUT #1:REC$
360 PRINT REC$
370 NEXT J
380 CLOSE #1
390 PRINT
400 PRINT "ORA AGGIUNGI ALTRI RECORD"
410 PRINT "NON RIAVVOLGERE IL NASTRO!"
420 PRINT "PER FINIRE SCRIVI FINE"
430 OPEN #1:"CS1",SEQUENTIAL,OUTPUT,INTE
RNAL,FIXED
440 PRINT
450 PRINT "SCRIVI UN RECORD"
460 INPUT REC$
470 IF REC$="FINE" THEN 510
480 PRINT #1:REC$
490 TOT=TOT+1
500 GOTO 440
510 CLOSE #1
520 PRINT
530 PRINT "NUMERO TOTALE DEI RECORD:"
540 PRINT TOT
550 PRINT
560 GOTO 200
```

BA.SE s.n.c.

SOFTWARE HOUSE - Casella Postale 4
13055 - Occhieppo Inferiore (VC)
Tel. 015/592730

SOFTWARE PER
COMMODORE 64
(disco o cassetta)

NOVITA'

COMBINATORE TELEFONICO - completo di cavi, linee adatte per collegamento alla linea e al Commodore 64, oppure VIC 20 (16 Kbyte). Fornito di software per disco o cassetta permette la ripetizione automatica del numero. Completo di archivio, numeri e indirizzi, consente di chiamare qualunque utente in memoria dato il nome o il codice. Non necessita di alcuna modifica all'impianto. Specificare se disco o cassetta. L. 75.000
CODICE 3 - Gioco di simulazione medica L. 30.000
ALTO MEDIOEVO - Simulazione economica L. 30.000
ATOMO - Simulazione centrale atomica L. 30.000
BIORITMI + HIDDEN CODE L. 30.000
TORRE DI HANOI + OTHELLO L. 30.000
BLACK JACK L. 30.000
DATA BASE SORG. - Sorgente per gestione archivi (solo dischetto) L. 50.000

SOFTWARE PER **TI 99/4A** (solo cassetta)

TILANDIA - Simulazione economica L. 25.000
SUPERBOWL - Simulazione football americano L. 25.000
SPACE TI/99 - In caccia fra le stelle L. 25.000
POKER + ROULETTE L. 25.000
LABIRINTO 3-D L. 25.000
PYRAMID - Adventure nella piramide maledetta L. 25.000
MEM L. 25.000
CONTABILITA' CASALINGA (pratica gestione) L. 50.000

SOFTWARE PER ATARI (solo cassetta)

ALTO MEDIOEVO - Simulazione economica L. 25.000

A vostra disposizione per software su misura per:

COMMODORE VIC 20 — OLIVETTI M 20
COMMODORE C-64 — IBM PC
OLIVETTI M 10 — IBM XT

Spedire in busta chiusa a:
BA.SE s.n.c. - Casella Postale 4 - 13055 Occhieppo inf. (VC)

Nome e Cognome _____

Indirizzo _____

Cap. _____ Città _____ Provincia _____

VIC 20 ☐ C 64 ☐ IBM PC ☐ M10 ☐ M20 ☐ Atari ☐ Texas ☐

Ordino n° _____ ☐ Disco ☐ Cassetta _____

Ordino n° _____ ☐ Disco ☐ Cassetta _____

Per un totale di Lire _____

Pagamento ☐ Allegato assegno non trasf. sped. celere
☐ Contro assegno + spese postali

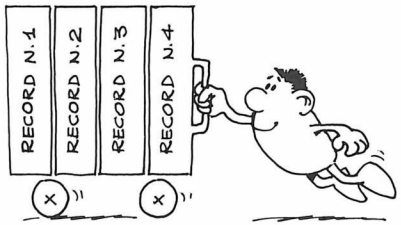
PERSONAL SOFTWARE

Apertura di un file nel modo APPEND

Il vantaggio di tutto ciò consiste nel poter trattare file molto estesi, in quanto nella memoria centrale non viene caricato che un record alla volta.

Esaminiamo brevemente la routine ed il modo di utilizzarla.

Le linee 130-160 scrivono un unico record per inizializzare il file. Per aggiungere nuovi record bisogna rispondere affermativamente alla richiesta di linea 200 ed indicare il numero di record già scritti. Inizialmente tale numero è 1, poiché è presente un unico record consistente nella stringa "RECORD N. 1". L'istruzione 310 apre il file nel modo INPUT e provoca sul video il noto messaggio ★ REWIND CASSETTE TAPE ...



Dopo la lettura del record il file viene chiuso ed appare il messaggio ★ PRESS CASSETTE STOP ... È il momento di aggiungere nuovi record. Il file viene aperto nuovamente nel modo OUTPUT, ma questa volta non si deve riavvolgere il nastro perché la nuova registrazione dovrà iniziare esattamente là dove è terminata la precedente.

Per provare si potranno scrivere record non dissimili dal primo, cioè: RECORD N. 2, RECORD N. 3, ecc.. Per terminare è previsto che si debba scrivere FINE ed ovviamente questa parola convenzionale non viene registrata su nastro, ma indica semplicemente al programma che il file deve essere chiuso.

Alla linea 490 la variabile TOT contiene il numero totale dei record scritti, cioè quelli preesistenti più quelli aggiunti; tale numero, come abbiamo già detto, va annotato accanto a quello che indica il punto di inizio del file.

Il programma, essendo scritto a semplice titolo d'esempio, è stato strutturato in modo ciclico. La linea 560 rinvia all'istruzione 200, dalla quale è possibile ripetere da capo le operazioni per aggiungere nuovi record ai precedenti.

I gruppi di informazioni scritti di volta in volta sono separati solo da un gap (cioè un intervallo vuoto sul nastro), un po' più lungo di quelli lasciati normalmente, che non disturba assolutamente la lettura sequenziale dell'intero file. ■

PERSONAL SOFTWARE

Atterraggio

Un programma ricreativo e istruttivo per Sharp MZ 700, C 64 e TI99

di *Martino Sangiorgio*

Nato come esercitazione pratica su diversi elaboratori, seguita a un corso teorico di programmazione in BASIC, il programma risente molto della necessità di essere, nel contempo, estremamente chiaro e adattabile a tutti gli elaboratori utilizzati durante il corso (in particolare, oltre allo Sharp MZ 700, erano presenti Texas TI99/4A, Commodore 64, Sinclair ZX Spectrum). Questo giustifica la lunghezza del listato; il tempo di digitazione è comunque compensato all'atto dell'utilizzo di questo appassionante gioco.

Funzionamento del programma

All'inizio viene emesso un menu per la scelta del pianeta su cui atterrare. I pianeti previsti sono i seguenti:

1 = Luna;
2 = Marte;
3 = Venere;
4 = Saturno.

Poiché la gravità dei pianeti elencati è via via sempre maggiore, anche il livello di difficoltà è crescente. A richiesta è possibile ottenere a video le informazioni necessarie al corretto uso del programma. Si consiglia, almeno la prima volta che si esegue il programma, di far uso di questa opzione; in seguito potrà non servire più.

Il programma evidenzia quindi sul video i valori iniziali della velocità dell'astronave (in km/h), dell'altezza di questa rispetto alla superficie del pianeta (in m) e del carburante a disposizione per l'atterraggio (in litri). Viene poi formulata la domanda: "Volete accendere i retrorazzi?". Se la risposta è negativa (si digita cioè "N"), l'astronave precipita in caduta libera verso il pianeta per un tempo di 5 secondi. La successiva maschera video mostrerà quindi i nuovi valori di altezza e velocità modificati in base alle leggi fisiche della caduta dei gravi (vedere "caduta libera"). La quantità di carburante resterà, naturalmente, invariata.

Se la risposta alla domanda precedente è positiva (viene cioè digitato "S"), due ulteriori domande vengono poste al giocatore: quanti retrorazzi si vogliono accendere e per quanto tempo.

L'astronave ha a disposizione 5 retrorazzi, ognuno dei quali consuma 150 litri di carburante ogni secondo. Per consentire la frenata è possibile quindi accendere da 1 a 5 retrorazzi, per un tempo variabile (in base alle necessità) da 0.1 secondi a 99,9 secondi. Naturalmente, ad ogni accensione di retrorazzi verrà

Listato 1. *Il programma Atterraggio.*

```
10 REM *------------------------------*
20 REM *                              *
30 REM *   ATTERRAGGIO SUI PIANETI    *
40 REM *                              *
50 REM *PERSONAL COMPUTER SHARP MZ700*
60 REM *                              *
70 REM *------------------------------*
80 REM  S1 = SCELTA DEL PIANETA
90 REM  I$ = SI/NO ISTRUZIONI
100 REM  G  = GRAVITA DEL PIANETA
110 REM  R$ = SI/NO RETRORAZZI
120 REM  S2 = SWITCH DI FINE PROGRAMMA
130 REM H0 = ALTEZZA INIZIALE (in metri)
140 REM V0 = VELOCITA' INIZIALE (in
            km/h)
150 REM C0 = CARBURANTE INIZIALE(in li-
            tri)
160 REM M0 = MASSA INIZIALE ASTRONAVE
170 REM M1 = MASSA DEL CARBURANTE CON-
            SUMATO OGNI SECONDO (in kg)
180 REM V1 = VELOCITA' DI USCITA DEL
            CARBURANTE DAI RETRORAZZI
            (in m/s)
190 REM V  = VELOCITA' RAGGIUNTA(in
            km/h)
200 REM S  = SPAZIO PERCORSO (in metri)
210 REM C  = CARBURANTE RIMASTO(in li-
            tri)
220 REM H  = ALTEZZA RAGGIUNTA(in metri)
230 REM M  = MASSA RESIDUA DELL'ASTRO-
            NAVE(in Kg)
240 REM M9 = MASSA CONSUMATA (in Kg)
250 REM N  = NUMERO DI RETRORAZZI ACCESI
260 REM T  = TEMPO DI ACCENSIONE CONSI-
            DERATO (in sec.)
270 REM A  = ACCELERAZIONE RISULTANTE
            in m/sec.*sec.)
280 REM V2 = VELOCITA' INIZIALE (in m/s)
290 REM V3 = VELOCITA' RAGGIUNTA (in
            m/s)
300 REM * Scelta del pianeta *
```

Segue listato Atterraggio.

```
310 REM        scelta del pianeta
320 GOSUB 630
330 IF S1=5 GOTO 4130
340 REM * Richiesta istruzioni *
350 GOSUB 870
360 IF I$="N" GOTO 390
370 REM * Stampa istruzioni *
380 GOSUB 970
390 REM * Inizializzazione variabili *
400 GOSUB 1410
410 ON S1 GOTO 420,440,460,480
420 G=1.6
430 GOTO 490
440 G=3.7
450 GOTO 490
460 G=8.7
470 GOTO 490
480 G=11.3
490 REM * Stampa maschera centrale *
500 GOSUB 1510
510 REM * Richiesta accensione retrorazz
i *
520 GOSUB 1650
530 IF R$="N"GOTO 570
540 REM * Accensione retrorazzi *
550 GOSUB 1870
560 GOTO 590
570 REM * Caduta libera *
580 GOSUB 2790
590 IF S2=1 GOTO 320
600 REM * Ricalcolo variabili *
610 GOSUB 3210
620 GOTO 490
630 REM *--------------------------*
640 REM *     Scelta del pianeta     *
650 REM *--------------------------*
660 COLOR ,,0,6
670 PRINT "■":PRINT:PRINT
680 PRINT " Con un'astronave dovete atte
rrare su"
690 PRINT " uno dei seguenti pianeti:"
700 PRINT
710 PRINT TAB(14);"1 = LUNA"
720 PRINT
730 PRINT TAB(14);"2 = MARTE"
740 PRINT
750 PRINT TAB(14);"3 = VENERE"
760 PRINT
770 PRINT TAB(14);"4 = SATURNO"
780 PRINT
790 PRINT TAB(14);"5 = FINE PROGRAMMA"
800 PRINT
```

consumato del carburante, per cui si dovrà fare attenzione a non restare a secco. Verranno quindi riproposti a video i nuovi valori di altezza, velocità e quantità di carburante, calcolati dalla routine "accensione retrorazzi". Se il nuovo valore di velocità proposto fosse negativo, significa che sono stati accesi troppi retrorazzi per un tempo eccessivo, per cui l'astronave, invece di scendere sul pianeta, se ne sta allontanando. Ciò può succedere di frequente sui pianeti a bassa gravità (per esempio la Luna); difficilmente capiterà sui pianeti a gravità maggiore. Se l'astronave arriva alla superficie del pianeta con una velocità inferiore ai 9 km/h, vuol dire che l'atterraggio si è felicemente concluso ed una opportuna maschera video si congratulerà col giocatore.

Se l'astronave arriva alla superficie del pianeta con una velocità superiore ai 9 km/h, verrà emessa una maschera per evidenziare il disastro; verranno anche visualizzati il valore della velocità all'atto dell'impatto con la superficie e la quantità di carburante rimasta. Se l'astronave rimane senza carburante prima ancora di aver potuto raggiungere la superficie del pianeta, verrà emessa una maschera che evidenzia a quanti metri di altezza è avvenuto il fatto e la velocità con cui l'astronave è precipitata poi sulla superficie del pianeta.

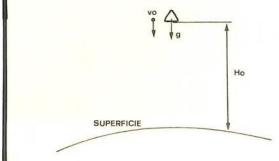


Figura 1. *Situazione iniziale.*

Caduta libera

La situazione iniziale dell'astronave è evidenziata dalla figura 1. Le leggi fisiche che regolano il moto di oggetti in caduta libera in un campo gravitazionale sono le seguenti:

$v = vo + gt$

$s = vot + {}^1/_2\, gt^2$

dove:

vo è la velocità iniziale (in m/sec);
v è la velocità dopo il tempo t (in m/sec);

g è l'accelerazione di gravità (in m/sec · sec);
t è il tempo preso in considerazione;
s è lo spazio percorso nel tempo t (in m).

In questo caso vo è dato dalla variabile Vo, v dalla variabile V, g dalla variabile G (impostata inizialmente in dipendenza del pianeta scelto), t è dato dalla variabile T, impostata inizialmente al valore 5 e s è dato dalla variabile S. Le unità di misura utilizzate vanno bene, tranne quelle della velocità. Si dovrà passare da km/h a m/sec con la relazione:
V2 = Vo/3,6.
Il nuovo valore di velocità sarà dato da:
V3 = V2 + G ★ T
che, ritrasformato in km/h, diventa:
V = V3 ★ 3,6
È possibile ora calcolare lo spazio percorso:
S = (V2 ★ T) + ((G ★ T²) ★ 0,5)
Si possono ora presentare tre casi:
a) S = Ho (figura 2)
In questo caso, se la velocità V è compresa tra −9 e +9 km/h, vuol dire che l'astronave è felicemente atterrata e verrà proposta la maschera "BRAVO", altrimenti l'astronave è atterrata ad una velocità troppo elevata, per cui verrà proposta la maschera "DISASTRO", che evidenzia i valori di V per la velocità e Co per il carburante.
b) S < Ho (figura 3)
In questo caso l'astronave non è ancora atterrata, per cui verranno eseguite nell'ordine:
H = Ho − S
Ho = H
Vo = V
e con questi nuovi valori di Ho e Vo (Co è rimasto invariato) verrà riproposta la maschera centrale.
c) S > Ho (figura 4)
Vuol dire che l'astronave ha superato la superficie del pianeta e si è inoltrata sottoterra. Sembrerebbe un disastro, ma con un tempo inferiore ai 5 sec avrebbe forse potuto atterrare (condizione S = Ho con velocità compresa tra −9 e +9 km/h). Si valuta la velocità all'atto dell'impatto sulla superficie eseguendo nell'ordine:

$$T = \frac{-V_2 + \sqrt{V_2^2 + 2GHo}}{G}$$

V3 = V2 + GT
V = V3 ★ 3,6

Se ora V sarà compreso tra −9 e +9, verrà proposta la maschera "BRAVO", altrimenti verrà stampata la maschera "DISASTRO", evidenziando i valori di V e Co.

Accensione retrorazzi

Il numero dei retrorazzi da considerare, letto da video, viene posto nella variabile N. Il tempo di accensione, anch'esso letto da video, viene posto nella variabile T. La quantità di carburante consumato al secondo da ogni retrorazzo (150 litri) è registrata nella variabile M1, mentre in Co è contenuta la quantità di carburante rimasta.
Se ora si avrà:
(M1 ★ N ★ T) ⩾ Co
significa che l'attuale accensione dei retrorazzi consuma tutto il carburante rimasto; in tal caso verrà eseguita la routine "carburante terminato".
In caso contrario applicheremo le leggi del moto uniformemente accelerato:
V = Vo + at
s = vot + ½ a t²
In questo caso l'accelerazione (a) è la risultante tra la gravità (diretta verso il pianeta) e quella data dai retrorazzi accesi (diretta in senso opposto); essa può anche diventare negativa, il che sta a significare che la potenza dei retrorazzi predomina sulla gravità e l'astronave tende quindi ad allontanarsi dal pianeta. La formula da applicare è la seguente (derivabile dalla fondamentale F = Fg ⊖ Fr) (vedere figura 5).

$$a = G + \frac{\Delta m}{M\Delta t} \cdot vr$$

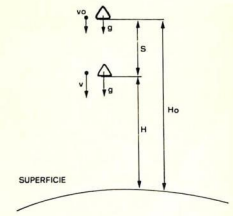


Figura 2. *L'astronave è arrivata sulla superficie (S = Ho).*
Se v ⩽ 9 km/h, l'atterraggio è concluso felicemente, altrimenti si è verificato un disastro.

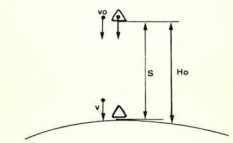


Figura 3. *L'astronave non è ancora atterrata (S<Ho).*

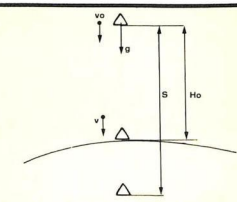


Figura 4. *Situazione in cui, nel tempo considerato, l'astronave percorrerebbe uno spazio S superiore alla distanza Ho dalla superficie (cioè S > Ho). Poiché tale caso è improbabile, bisogna riconsiderare le formule fisiche del moto, ponendo S = Ho e calcolando il valore della velocità in questo caso (cioè all'atto dell'impatto con la superficie). Se v ⩽ 9 km/h, l'atterraggio è felicemente concluso, altrimenti si è verificato un disastro.*

Segue listato Atterraggio.

```
810 PRINT
820 PRINT " ESPRIMERE SCELTA "
830 GET S1
840 IF S1<=0 GOTO 830
850 IF S1> 5 GOTO 830
860 RETURN
870 REM *----------------------------*
880 REM *Richiesta SI o NO istruzioni*
890 REM *----------------------------*
900 REM cancella schermo :PRINT "■"
910 PRINT "VOLETE ULTERIORI ISTRUZIONI ?
 (S/N) "
920 GET I$
930 IF I$="S" GOTO 960
940 IF I$="N" GOTO 960
950 GOTO 920
960 RETURN
970 REM *----------------------------*
980 REM *     Stampa istruzioni      *
990 REM *----------------------------*
1000 REM cancella schermo :PRINT "■"
1010 PRINT
1020 PRINT " Se non rallentata da retror
azzi, la"
1030 PRINT " astronave precipita sul pia
neta scelto"
1040 PRINT " seguendo la legge della cad
uta dei"
1050 PRINT " gravi;aumentando man mano v
elocita'e'
1060 PRINT " spazio percorso in base all
a gravita'"
1070 PRINT " del pianeta stesso. In tal c
aso vengono"
1080 PRINT " determinati i nuovi valori
di altezza"
1090 PRINT " e velocita' ogni 5 secondi.
Per consen-"
1100 PRINT " tire la frenata e quindi l'
atterrag-"
1110 PRINT " gio, e' possibile accendere
 da 1 a 5 "
1120 PRINT " retrorazzi per un tempo pre
stabilito,"
1130 PRINT " variabile tra 0.1 e 99.99 s
econdi."
1140 PRINT
1150 PRINT
1160 PRINT " BATTERE UN TASTO"
1170 GET K$:IF K$="" GOTO 1170
1180 PRINT "■"
1190 PRINT
1200 PRINT " I retrorazzi consumano (nat
uralmen-"
1210 PRINT " te), ad ogni accensione, un
a parte "
1220 PRINT " del carburante: ottenzione
quindi a "
1230 PRINT " non rimanerne senza. Il con
sumo di "
1240 PRINT " carburante e' costante nell
'unita' "
```

dove:
Δm = perdita di massa dall'astronave (cioè carburante consumato);
M = massa dell'astronave;
Δt = intervallo di tempo considerato;
vr = velocità di uscita del carburante dai retrorazzi.
Il valore di g è contenuto nella variabile G (impostata inizialmente in dipendenza del pianeta scelto), Δm è dato dal prodotto M1 ★ N ★ T, vr è dato dalla variabile V1, M è dato dalla variabile Mo e Δt dalla variabile T, mentre il valore di a verrà posto nella variabile A. Si avrà quindi:

$$A = G - \frac{V1 \star M1 \star N}{Mo}$$ (v. fig. 6)

Si potranno quindi calcolare i nuovi valori di velocità, altezza e carburante, utilizzando, nell'ordine, le seguenti formule:

V2 = Vo/3,6
V3 = V₂ + A★T
V = V₃ ★ 3,6
S = (V₂ ★ T) + ((A ★ T ↑ 2) ★ 0,5)
C = Co − (M1 ★ N ★ T)

Si propongono ora di nuovo i seguenti tre casi:
a) S = Ho (figura 2)
Se V è compresa tra −9 e +9 km/h, verrà evidenziata la maschera "BRAVO", altrimenti verrà proposta la maschera "DISASTRO", evi-

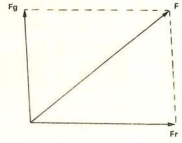


Figura 5. *Risultato della composizione di due forze aventi direzione qualsiasi (F = Fg ⊕ Fr).*

Figura 6. *Risultato della composizione di due forze aventi direzioni opposte. La forza risultante è la differenza tra le due (F = Fg − Fr).*

Segue listato Atterraggio.

```
1250 PRINT " di tempo e non dipende dall
a gravi- "
1260 PRINT " ta' del pianeta scelto. I n
uovi va- "
1270 PRINT " lori dell'altezza e della v
elocita'"
1280 PRINT " vengono determinati dopo il
 tempo "
1290 PRINT " che avete stabilito per l'a
ccensio-"
1300 PRINT " ne dei retrorazzi. "
1310 PRINT
1320 PRINT
1330 PRINT "        AUGURI E BUONA FORTUN
A ☺  "
1340 PRINT
1350 PRINT
1360 PRINT " BATTERE UN TASTO"
1370 GET K$:IF K$="" GOTO 1370
1380 PRINT "■"
1390 RETURN
1400 REM *----------------------------*
1410 REM * Inizializzazione variabili *
1420 REM *----------------------------*
1430 V1=1000
1440 M1=150
1450 H0=INT(10001*RND(1))+30000
1460 V0=INT(41*RND(1))+140
1470 C0=H0+INT(4001*RND(1))+4000
1480 M0=C0+INT(2001*RND(1))+10000
1490 RETURN
1500 REM *--------------------------*
1510 REM * Stampa maschera centrale *
1520 REM *--------------------------*
1530 PRINT "■"
1540 PRINT
1550 PRINT
1560 PRINT TAB(4);"ALTEZZA";TAB(17);H0;T
AB(24);"metri"
1570 PRINT
1580 PRINT TAB(4);"VELOCITA'";TAB(17);V0
;TAB(24);"Km/h"
1590 PRINT

1600 PRINT TAB(4);"CARBURANTE";TAB(17);C
0;TAB(24);"litri"
1610 PRINT
1620 PRINT
1630 RETURN
1640 REM *----------------------*
1650 REM * Richiesta accensione *
1660 REM *      retrorazzi      *
1670 REM *----------------------*
1680 PRINT " VOLETE ACCENDERE I RETRORAZ
ZI ?(S/N)"
```

nuovidea

Riviste firmate JACKSON

Bit
La prima Rivista europea di personal computer, software e accessori. Con test, novità, analisi del mercato...
11 numeri all'anno.
L. 4.000 a numero
Abbonamento: solo **L. 35.000**

PERSONAL SOFTWARE
Aspetti e problemi del software per personal computer, programmi, giochi e sistemi operativi.
10 numeri all'anno.
L. 3.500 a numero
Abbonamento: solo **L. 28.000**

INFORMATICA
La Rivista professionale per chi si occupa di sistemi: dai microcomputer ai mini, ai supermini, ai mainframe. Con notizie in anteprima dall'America.
11 numeri all'anno.
L. 3.000 a numero
Abbonamento: solo **L. 27.000**

VIDEO GIOCHI
La guida indiscussa al fantastico mondo dei videogames. La più eccitante, divertente, istruttiva rassegna del settore.
11 numeri all'anno.
L. 3.000 a numero
Abbonamento: solo **L. 25.000**

elettronica oggi
Da sedici anni il punto di riferimento più qualificato per chi voglia aggiornarsi su prodotti, applicazioni, tecnologie, mercati, in Italia e all'estero.
11 numeri all'anno.
L. 3.500 a numero
Abbonamento: solo **L. 31.000**

l'Elettronica
Quindicinale di politica industriale, componentistica, informatica e telecomunicazioni per uomini di marketing, responsabili acquisti, manager di settore.
22 numeri all'anno.
L. 2.500 a numero
Abbonamento: solo **L. 44.000**

AUTOMAZIONE
Un'aggiornatissima panoramica delle nuove tecnologie microelettroniche e informatiche applicate all'automazione industriale.
11 numeri all'anno.
L. 3.000 a numero
Abbonamento: solo **L. 26.000**

telecomunicazioni
Le frontiere aperte dalla telematica, le telecomunicazioni professionali in tutti i loro sottosettori.
8 numeri all'anno.
L. 3.500 a numero
Abbonamento: solo **L. 22.000**

elektor
Il mensile di elettronica venduto in mezzo milione di copie e redatto in 7 lingue. Con articoli su: applicazioni, progettazioni, sperimentazioni, invenzioni.
10 numeri all'anno.
L. 3.000 a numero
'Numero doppio' **L. 6.000**
Abbonamento: solo **L. 29.000**

strumenti MUSICALI
Il mondo delle 7 note in versione... elettronica. Con test strumentali, novità e analisi del mercato, servizi speciali.
10 numeri all'anno.
L. 3.000 a numero
Abbonamento: solo **L. 24.000**

Quando l'informazione fa testo

In busta chiusa inviate questo coupon a:
Gruppo Editoriale Jackson - via Rosellini, 12 - 20124 Mi

☐ Desidero ricevere GRATIS un numero della Rivista _____
(allego L. 1.000 in francobolli per contributo spese di spedizione)

☐ Inviatemi GRATIS il Catalogo della Biblioteca JACKSON (allego L. 1.000 in francobolli per contributo spese di spedizione)

Nome _____ Cognome _____

via _____

CAP _____ Città _____

Segue listato Atterraggio.

```
1690 GET R$
1700 IF R$="N" GOTO 1850
1710 IF R$="S" GOTO 1730
1720 GOTO 1690
1730 REM * Richiesta numero retrorazzi *
7090 PRINT
1740 PRINT
1750 PRINT "QUANTI ? "
1760 GET N
1770 IF N<1 GOTO 1760
1780 IF N>5 GOTO 1760
1790 REM * Richiesta tempo di accensione
       retrorazzi *
1800 PRINT
1810 INPUT "PER QUANTO TEMPO ? ";T
1820 IF (T>=.1)*(T<=99.9) GOTO 1850
1830 GOSUB 1510
1840 GOTO 1680
1850 RETURN
1860 REM *----------------------*
1870 REM * Accensione retrorazzi *
1880 REM *----------------------*
1890 M9=INT(M1*N*T)
1900 IF M9>=C0 GOTO 2280
1910 A=G-(V1*M1*N)/M0
1920 V2=V0/3.6
1930 V3=V2+A*T
1940 V=V3*3.6
1950 S=(V2*T)+((A*T↑2)*.5)
1960 S=INT(S)
1970 V=INT(V)
1980 C=C0-M9
1990 IF S=H0 GOTO 2040
2000 IF S>H0 GOTO 2140
2010 REM * Non ancora atterrata *
2020 S2=0
2030 GOTO 2770
2040 REM * Arrivata alla superficie *
2050 S2=1
2060 IF V<-9 GOTO 2110
2070 IF V>9 GOTO 2110
2080 REM * Stampa maschera 'BRAVO' *
2090 GOSUB 3520
2100 GOTO 2770
2110 REM * Stampa maschera 'DISASTRO' *
2120 GOSUB 3300
2130 GOTO 2770
2140 REM * Oltre la superficie *
2150 S2=1
2160 T=(-V2+SQR(V2↑2+2*A*H0))/A
2170 V3=V2+A*T
2180 V=V3*3.6
2190 V=INT(V)
2200 IF V<-9 GOTO 2250
2210 IF V>9 GOTO 2250

2220 REM * Stampa maschera 'BRAVO' *
2230 GOSUB 3520
2240 GOTO 2770
2250 REM * Stampa maschera 'DISASTRO' *
2260 GOSUB 3300
2270 GOTO 2770
2280 REM * Carburante terminato *
```

denziando i valori di V e C.

b) S < Ho (figura 3)

L'astronave non è ancora atterrata, per cui si calcoleranno i nuovi valori iniziali, eseguendo nell'ordine:

H = Ho − S
M = Mo − (M1 ★ N ★ T)
Ho = H
Vo = V
Co = C
Mo = M

Con questi nuovi valori verrà riproposta la maschera centrale.

c) S > Ho (figura 4)

L'astronave è andata oltre la superficie del pianeta. Bisogna valutare se, al momento dell'impatto, aveva una velocità accettabile. Come visto in precedenza, verranno eseguite nell'ordine:

$$T = \frac{-V2 + \sqrt{V_2^2 + 2 \star A \star Ho}}{A}$$

V3 = V2 + A ★ T
V = V3 ★ 3,6

Se ora V è compresa tra −9 e +9 km/h, verrà stampata la maschera "BRAVO", altrimenti verrà proposta la maschera "DISASTRO", evidenziando i valori di V e C.

Carburante terminato

Con la manovra in corso l'astronave esaurisce il carburante, per cui bisogna verificare se i valori introdotti permettono l'atterraggio.

Verranno eseguiti quindi, nell'ordine:

V2 = Vo/3,6
A = G − (Co ★ V1/Mo ★ T)
V3 = V2 + A ★ T
V = V3 ★ 3,6
S = V2 ★ T + $^1/_2$ A ★ T^2

Si ripropongono ora di nuovo i tre casi:

a) S = Ho (figura 2)

Se V è compreso tra −9 e +9 km/h, verrà stampata la maschera "BRAVO", altrimenti verrà visualizzata la maschera "DISASTRO" con i valori di V per la velocità e con zero per il carburante.

b) S < Ho (figura 3).

L'astronave non è ancora atterrata, perciò, essendo rimasta senza carburante, cadrà dall'ultima altezza raggiunta sulla superficie del pianeta, seguendo le leggi già viste in precedenza nella routine "CADUTA LIBERA", schiantandosi ad una velocità che viene calcolata eseguendo nell'ordine:

H = Ho − S
V2 = V/3,6

$$T = \frac{-V2 + \sqrt{V2^2 + 2 \star G \star H}}{G}$$

V3 = V2 + G ★ T

Atterraggio

Segue listato Atterraggio.

```
2290 V2=V0/3.6
2300 A=G-(C0*V1)/(M0*T)
2310 V3=V2+A*T
2320 V=V3*3.6
2330 S=(V2*T)+((A*T↑2)*.5)
2340 S=INT(S)
2350 V=INT(V)
2360 C=0
2370 S2=1
2380 IF S=H0 GOTO 2550
2390 IF S>H0 GOTO 2650
2400 REM * Non ancora atterrata - caduta
       libera - *
2410 H=H0-S
2420 V2=V/3.6
2430 T=(-V2+SQR(V2↑2+2*G*H))/G
2440 V3=V2+G*T
2450 V=V3*3.6
2460 V=INT(V)
2470 IF V<-9 GOTO 2520
2480 IF V>9 GOTO 2520
2490 REM * Stampa maschera 'BRAVO' *
2500 GOSUB 3520
2510 GOTO 2770
2520 REM * Stampa maschera 'PECCATO' *
2530 GOSUB 3740
2540 GOTO 2770
2550 REM * Arrivata alla superficie *
2560 IF V<-9 GOTO 2610
2570 IF V>9 GOTO 2610
2580 REM * Stampa maschera 'BRAVO' *
2590 GOSUB 3520
2600 GOTO 2770
2610 REM * Stampa maschera 'DISASTRO' *
2620 GOSUB 3300
2630 GOTO 2770
2640 REM * Oltre la superficie *
2650 T=(-V2+SQR(V2↑2+2*A*H0))/A
2660 V3=V2+A*T
2670 V=V3*3.6
2680 V=INT(V)
2690 IF V<-9 GOTO 2740
2700 IF V>9 GOTO 2740
2710 REM * Stampa maschera 'BRAVO' *
2720 GOSUB 3520
2730 GOTO 2770
2740 REM * Stampa maschera 'DISASTRO' *
2750 GOSUB 3300
2760 GOTO 2770
2770 RETURN
2780 REM *--------------*
2790 REM * Caduta libera *
2800 REM *--------------*
2810 T=5
2820 V2=V0/3.6
2830 V3=V2+G*T
2840 V=V3*3.6
2850 M9=0
2860 C=C0

2870 S=(V2*T)+((G*T↑2)*.5)
2880 S=INT(S)
2890 V=INT(V)
2900 IF S=H0 GOTO 2950
2910 IF S>H0 GOTO 3050
2920 REM * Non ancora atterrata *
2930 S2=0
2940 GOTO 3190
2950 REM * Arrivata alla superficie *
2960 S2=1
2970 IF V<-9 GOTO 3020
2980 IF V>9 GOTO 3020
2990 REM * Stampa maschera 'BRAVO' *
3000 GOSUB 3520
3010 GOTO 3190
3020 REM stampa maschera 'DISASTRO'
3030 GOSUB 3300
3040 GOTO 3190
3050 REM * Oltre la superficie *
3060 S2=1
3070 T=(-V2+SQR(V2↑2+2*G*H0))/G
3080 V3=V2+G*T
3090 V=V3*3.6
3100 V=INT(V)
3110 IF V<-9 GOTO 3160
3120 IF V>9 GOTO 3160
3130 REM * Stampa maschera 'BRAVO' *
3140 GOSUB 3520
3150 GOTO 3190
```

Tabella 1. *Elenco delle principali variabili.*

Ho = altezza iniziale (m)
Vo = velocità iniziale (km/h)
Co = carburante iniziale (l)
Mo = massa iniziale dell'astronave (kg)
V = velocità raggiunta (km/h)
S = spazio percorso (m)
C = carburante rimasto (l)
N = numero retrorazzi accesi
H = altezza raggiunta (m)
T = tempo considerato (sec)
M = massa residua dell'astronave (kg)
M1 = massa del carburante consumato ogni secondo (kg)
G = gravità del pianeta scelto (m/sec · sec)
A = accelerazione risultante (m/sec · sec)
V1 = velocità di uscita del carburante dai retrorazzi (m/sec)

Tabella 2. *Valori delle accelerazioni di gravità dei diversi pianeti*

Luna	G = 1.6
Marte	G = 3.7
Venere	G = 8.7
Saturno	G = 11.3

Atterraggio

Segue listato Atterraggio.

```
3160 REM * Stampa maschera 'DISASTRO' *1
1400 GOSUB 16010
3170 GOSUB 3300
3180 GOTO 3190
3190 RETURN
3200 REM *---------------------*
3210 REM * ricalcolo variabili *
3220 REM *---------------------*
3230 H=H0-S
3240 M=M0-M9
3250 H0=H
3260 V0=V
3270 C0=C
3280 M0=M
3290 RETURN
3300 REM *--------------------------*
3310 REM *     MASCHERA DISASTRO    *
3320 REM *--------------------------*
3330 COLOR ,,7,0
3340 PRINT "@"
3350 PRINT
3360 PRINT
3370 PRINT "  D I S A S T R O ! ! "
3380 PRINT
3390 PRINT
3400 PRINT
3410 PRINT " Vi siete schiantati sul pia
neta ad una"
3420 PRINT " velocita' di  " ;V;TAB(23);
"Km/h"
3430 PRINT " avevate ancora";C;TAB(23);"
litri di carbu-"
3440 PRINT " rante."
3450 PRINT
3460 PRINT
3470 PRINT
3480 GOSUB 3980
```

```
3490 PRINT "BATTERE UN TASTO"
3500 GET K$:IF K$="" GOTO 3500
3510 RETURN
3520 REM *--------------------------*
3530 REM *       MASCHERA BRAVO     *
3540 REM *--------------------------*
3550 COLOR ,,7,0
3560 PRINT "@"
3570 PRINT
3580 PRINT
3590 PRINT "        B R A V O ! ! "
3600 PRINT
3610 PRINT
3620 PRINT
3630 PRINT " Con estrema perizia siete r
iuscito ad "
3640 PRINT " atterrare sul vostro pianet
o preferito."
3650 PRINT " Non affidatevi pero' troppo
 alla vostra"
3660 PRINT " buona sorte: riposatevi ben
e prima di"
3670 PRINT " ritentare."
3680 PRINT
3690 PRINT
3700 GOSUB 4050
3710 PRINT "BATTERE UN TASTO"
3720 GET K$:IF K$="" GOTO 3720
3730 RETURN
3740 REM *--------------------------*
3750 REM *     MASCHERA PECCATO     *
3760 REM *--------------------------*
3770 COLOR ,,7,0
3780 PRINT "@"
3790 PRINT
3800 PRINT
3810 PRINT "        P E C C A T O ! ! "
3820 PRINT
3830 PRINT
3840 PRINT
3850 PRINT " Forse avreste potuto farcel
a,ma siete"
3860 PRINT " rimasti senza carburante a"
;TAB(28);H
3870 PRINT " metri di altezza, per cui s
iete preci-"
```

$V = V3 \star 3{,}6$

A questo punto se V fosse ancora compresa tra +9 e −9 km/h, verrà proposta la maschera "BRAVO", altrimenti verrà emessa la maschera "PECCATO", evidenziando i valori di H per l'altezza e di V per la velocità.

c) $S > Ho$ (figura 4)

Si dovrà verificare la velocità al momento dell'impatto sulla superficie, applicando nell'ordine le formule:

$$T = \frac{-V2 + \sqrt{V2^2 + 2 \star A \star Ho}}{A}$$

$V3 = V2 + A \star T$

$V = V3 \star 3{,}6$

Se ora V è compresa tra −9 e +9 km/h, verrà proposta la maschera "BRAVO", altrimenti la maschera "DISASTRO" con i valori di V per la velocità e zero per il carburante.

Commenti al listato

Nel programma si è fatto un largo uso di commenti, per cui esso si auto-documenta molto bene.

10-290 Elenco delle variabili utilizzate.

300-620 Corpo del programma.

410-480 Impostazione del valore di G (gravità) in base al pianeta prescelto.

630-680 Routine "scelta del pianeta". Permette di determinare su quale pianeta si vuole effettuare l'atterraggio.

870-960 Routine "Richiesta SI - NO istruzioni". Stampa il messaggio "volete ulteriori istruzioni?" e pone la risposta ("S" o "N") nella variabile I$.

970-1390 Routine "Stampa istruzioni". Viene eseguita se I$ = "S".

Segue listato Atterraggio.

```
3880 PRINT " pitati sul pianeta ad una v
elocita' di"
3890 PRINT  V;TAB(9);"Km/h"
3900 PRINT
3910 PRINT
3920 PRINT
3930 GOSUB 3980
3940 PRINT "BATTERE UN TASTO"
3950 GET K$:IF K$="" GOTO 3950
3960 RETURN
3970 REM *---------------------*
3980 REM * Rumore del disastro *
3990 REM *---------------------*
4000 MM$="-G1R-GR-GR-#D9R1-FR-FR-FR-D9"
4010 TEMPO 6
4020 MUSIC MM$,MM$
4030 RETURN
```

```
4040 REM *-----------------*
4050 REM * Saluto musicale *
4060 REM *-----------------*

4070 MM$="A3":M1$="A5+#C3+D+E+#F+G+#F0+G
+#F4+E3+D+#CB":
4080 M2$="A3+D2R0+D1+E2+D+#C3B+#C7+#C3"
4090 M3$="A3+#C2R0+#C1+D2+#CB3A+D7+D3"
4100 TEMPO 3
4110 MUSIC MM$,M1$,M2$,M1$,M3$,M1$,M2$,M
1$,M3$
4120 RETURN
4130 REM *---------------------------*
4140 REM *           FINE            *
4150 REM *---------------------------*
4160 COLOR ,,7,1
4170 PRINT "▯"
4180 END
```

1400-1490 Routine "Inizializzazione variabili". Determina i valori iniziali di alcune variabili e precisamente:

- Vl = 1000 (m/sec)
- Ml = 150 (litri)
- Ho = a caso tra 30.000 e 40.000 (m)
- Vo = a caso tra 140 e 180 (km/h)
- Co = Ho + un numero a caso tra 4000 e 8000 (litri)
- Mo = Co + un numero a caso tra 10.000 e 12.000 (litri)

1500-1630 Routine "stampa maschera centrale". Visualizza a video i valori attuali di Ho (altezza), Vo (velocità) e Co (carburante).

1640-1850 Routine "Richiesta accensione retrorazzi". Richiede se si vogliono accendere dei retrorazzi e memorizza la risposta ("S" o "N") nella variabile R$. Se si è risposto "S", richiede il numero di retrorazzi, che memorizza nella variabile N, e il tempo di accensione degli stessi, che memorizza nella variabile T. Controlla anche che i valori digitati siano corretti ($5 \geqslant N \geqslant 1$ e $99.9 \geqslant T \geqslant 0.1$).

1860-2770 Routine "Accensione retrorazzi". Viene eseguita quando R$ = "S". Per il funzionamento vale quanto specificato nel testo. Contiene anche la routine "Carburante terminato" (2280-2770).

2780-3190 Routine "Caduta libera". Viene eseguita quando R$ = "N". Vale quanto specificato nel testo.

84 BIAS in Milan

MILANO QUARTIERE FIERA
Orario: 9,30 - 18,00
Ingresso: Porta Carlo Magno
e Porta Meccanica (MM Piazza Amendola)

MILAN TRADE FAIR PREMISES
Open: 9.30 a.m. - 6.00 p.m.
Entrance: Porta Carlo Magno
and Porta Meccanica (MM Piazza Amendola)

Tagliando per il visitatore

19 CONVEGNO MOSTRA INTERNAZIONALE DELL'AUTOMAZIONE STRUMENTAZIONE E MICROELETTRONICA
INTERNATIONAL AUTOMATION INSTRUMENTATION AND MICROELECTRONICS CONFERENCE AND EXHIBITION
29 Novembre - 4 Dicembre 1984
29th November - 4th December 1984

Invito offerto da
Free entrance ticket
offered by

GRUPPO EDITORIALE JACKSON
PERSONAL SOFTWARE

In concomitanza con 8ª RICH e MAC '84
In concomitance with 8th RICH and MAC '84

Compilare in stampatello in ogni sua parte.
Please fully fill in capital letters.

Cognome ..... (family name)
Nome ..... (christian name)
Ditta/Ente ..... (firm)
Via ..... (street)
Cap ..... (zip) Città ..... (town)
Prov. ..... (state)

Attenzione: COMPILARE ANCHE IL RETRO →

Atterraggio

3200-3290 Routine "Ricalcolo variabili". Aggiorna i valori di Ho, Vo, Co prima che vengano di nuovo stampati dalla maschera centrale. Aggiorna pure il valore di Mo.
3300-3510 Stampa una maschera a video in caso di disastro, evidenziando anche i valori di V e di C. Verrà anche generato un rumore opportuno.
3520-3730 Stampa una maschera a video di congratulazioni per l'atterraggio concluso felicemente. È accompagnata da una musichetta di saluto.
3740-3960 Stampa una maschera a video per il caso di completo consumo di carburante, evidenziando anche i valori di H e di V.
3970-4030 Genera un rumore che simula il disastro.
4040-4120 Genera una musichetta allegra per salutare l'avvenuto atterraggio.
4130-4180 Rimette l'elaboratore nelle condizioni di colore iniziali, pulisce il video e fa terminare il programma. ■

VARIANTI DA EFFETTUARE AL PROGRAMMA PER POTERLO ESEGUIRE SU TEXAS TI99/4A (IN TI BASIC)

1) Sostituire tutte le PRINT " [C] " con CALL CLEAR (funzione di pulizia del video)
2) Sostituire tutte le GOTO nelle istruzioni IF con delle THEN
3) Sostituire la linea 660 con le seguenti:
660 CALL SCREEN (12)
662 FOR I=1 TO 12
664 CALL COLOR (I,2,1)
666 NEXT I
4) Rivedere la lunghezza delle PRINT. Infatti il TI99/4A ha il video suddiviso in 32 colonne, mentre lo Sharp MZ-700 (e il CBM 64) hanno il video suddiviso in 40 colonne.
5) Modificare la linea 830 come segue:
830 CALL KEY (0,S1,ST)
6) Modificare la linea 920 come segue:
920 CALL KEY (0,I$,ST)
7) Sostituire la linea 1170 con le seguenti:
1170 CALL KEY (0,K$,ST)
1172 IF ST=0 THEN 1170
8) Sostituire la linea 1370 con le seguenti:
1370 CALL KEY (0,K$,ST)
1372 IF ST=0 THEN 1370
9) Sostituire le linee da 1450 a 1480 con le seguenti:
1450 H0=INT (RND★10001)+30000
1460 V0=INT (RND★41)+140
1470 C0=H0+INT(RND★4001)+4000
1480 M0=C0+INT(RND★2001)+10000
10) Sostituire la linea 1690 con la seguente:
1690 CALL KEY (0,R$,ST)
11) Sostituire la linea 1760 con la seguente:
1760 CALL KEY (0, N, ST)
12) Sostituire la linea 1810 con la seguente:
1810 INPUT "PER QUANTO TEMPO?":T
13) Sostituire la linea 3330 con le seguenti:
3330 CALL SCREEN (2)
3332 FOR I=1 TO 12
3334 CALL COLOR (I,16,1)
3336 NEXT I
14) Sostituire la linea 3500 con le seguenti:
3500 CALL KEY (0,K$,ST)
3502 IF ST=0 THEN 3500
15) Sostituire la linea 3550 con le seguenti:
3550 CALL SCREEN (2)
3552 FOR I=1 TO 12
3554 CALL COLOR (I,16,1)
3556 NEXT I
16) Sostituire la linea 3720 con le seguenti:
3720 CALL KEY (0,K$,ST)
3722 IF ST=0 THEN 3720
17) Sostituire la linea 3770 con le seguenti:
3770 CALL SCREEN (2)
3772 FOR I=1 TO 12
3774 CALL COLOR (I,16,1)
3776 NEXT I
18) Sostituire la linea 3950 con le seguenti:
3950 CALL KEY (0,K$,ST)
3952 IF ST=0 THEN 3950
19) Sostituire la routine "Rumore del disastro" con la seguente:
4000 FOR I=0 TO 30
4010 CALL SOUND (150,−5,1)
4020 NEXT I
4030 RETURN
20) Sostituire la routine "Saluto musicale" con la seguente:
4070 TONO=110
4080 FOR I=1 TO 10
4090 CALL SOUND (300,TONO,1)
4100 TONO=TONO+110
4110 NEXT I
4120 RETURN
oppure con altra musica voluta.

VARIANTI DA EFFETTUARE AL PROGRAMMA PER POTERLO ESEGUIRE SU COMMODORE 64

1) Sostituire la linea 660 con la seguente:
660 POKE 53280,7:POKE 53281,7: PRINT "[←]"
2) Sostituire tutte le PRINT " [C] " con PRINT " [●] " (funzione di pulizia del video)
3) Sostituire la linea 3330 con la seguente:
3330 POKE 53280,9:POKE 53281,0:PRINT " [E] "
4) Sostituire la linea 3550 con la seguente:
3550 POKE 53280,9:POKE 53281,0:PRINT " [E] "
5) Sostituire la linea 3770 con la seguente:
3770 POKE 53280,9:POKE 53281,0:PRINT " [E] "
6) Sostituire la routine "Rumore del disastro" con la seguente:
4000 V=54296:W=54276:A=54277:H=54273:L=54272
4010 FOR X=15 TO 0 STEP −1: POKE V,X: POKE W,129
4015 POKE A,15:POKE H,40:POKE L,200
4020 FOR I=1 TO 100:NEXT:NEXT
4025 POKE W,0:POKE A,0
4030 RETURN
7) Sostituire la routine "Saluto musicale" con la seguente:
4070 POKE 54296,15:POKE 54277,9:POKE 54276,17
4075 FOR I=1 TO 300:NEXT I
4080 READ A:READ B
4085 IF B= −1 THEN POKE 54273,0:POKE 54272,0:POKE 54296,0:GOTO 4120
4090 POKE 54273,A:POKE 54272,B
4095 POKE 54276,0:POKE 54276,17
4097 GOTO 4075
4100 DATA 17,37,19,63,21,154,22,227
4105 DATA 25,177,28,214,32,94,34,175
4110 DATA −1, −1
4120 RETURN

PERSONAL SOFTWARE

Ripartizione spese di riscaldamento

Un programma di tipo amministrativo. È richiesto il modulo TI Extended BASIC

di *Sergio Borsani*

Quando ho iniziato a scrivere questo programma pensavo solo di proporre un esempio sull'uso delle matrici a due dimensioni.

Quando mi sono accorto che con qualche aggiunta e qualche miglioramento poteva diventare operativo, in quattro ore per la stesura, più altre due o tre per il debugging e la messa a punto, ho portato a termine questo lavoro che non sarà la gioia di quanti concepiscono l'home computer come un compagno di giochi intellettuali (o di giochi soltanto) ma giungerà certamente gradito a quanti desiderano mettere il naso, e non solo il naso, nei conti che manda a casa l'amministratore. E perchè no, per le sue caratteristiche il programma potrà essere utilizzato proprio dall'amministratore!

Quale configurazione è richiesta?

È necessario il modulo SSS TI Extended BASIC. Ho scelto questa versione soprattutto per usufruire dei comandi DISPLAY AT, ACCEPT AT e PRINT USING che permettono di creare videate e tabulazioni molto meglio di quanto non si possa fare con il BASIC residente. La struttura multistatement (più istruzioni sulla stessa linea) rende più difficile la lettura e la trascrizione, perciò attenzione! Non è necessaria alcuna espansione di memoria, il programma occupa 7,3 Kbyte. La stampante non è strettamente neces-

RIPARTIZIONE SPESE DI RISCALDAMENTO AL 31/01/84.

GASOLIO	L.	4560000
ENEL	L.	134000
TOTALE	L.	4694000
DI CUI IL 20 % ... PER SPESE FISSE	L.	938800
E LE RESTANTI ... DA RIPARTIRE IN BASE AI CONSUMI	L.	3755200

n.	cognome e nome	sup. rad.	ore	ore * sup.	spese fisse	spese al consumo	TOTALE
1.	ROSSI G.	35	450	15750	139821	346386	486207
2.	ALBERTI M.	24	835	20040	95877	440734	536611
3.	VERDI E.	30	745	22350	119846	491536	611382
4.	BORSANI S.	30	935	28050	119846	616894	736740
5.	MORATTI A.	22	649	14278	87887	314011	401898
6.	GUIDI F.	38	852	32376	151805	712034	863839
7.	POLI P.	20	583	11660	79897	256435	336332
8.	BERGAMI L.	36	729	26244	143816	577175	720991
	TOTALI	235	5778	170748	938800	3755200	4694000

l'amministratore

Figura 1 - *Prospetto riassuntivo elaborato dal computer. La ripartizione viene fatta in modo convenzionale, presupponendo l'esistenza di contatori per rilevare le ore di consumo di ciascun appartamento. Una parte percentuale del totale, secondo le indicazioni dell'utente, viene ripartita come quota fissa mentre la rimanente è ripartita in base ai consumi.*

Ripartizione spese di riscaldamento

saria, chi ne è sprovvisto dovrà accontentarsi dell'output su video. Esso è del tutto simile alla copia su carta con la differenza che le 28 colonne del TI99/4A non consentono un'ampia tabulazione e si deve osservare il prospetto finale suddiviso in tre sezioni. Su carta, invece, il risultato è quello illustrato nella figura 1.

In che modo viene effettuata la ripartizione

Non tutti i condominii hanno le stesse esigenze in fatto di ripartizione. Il programma prevede che si tenga conto sia della superficie radiante di ogni appartamento sia delle ore di consumo di ogni condòmino (molte abitazioni sono sprovviste di termostato e di contatore delle ore di consumo individuale).

La spesa totale, alla quale possono concorrere più voci, viene divisa in due parti delle quali la prima verrà suddivisa in base alle superfici radianti per costituire le quote fisse e la seconda in base ai consumi. Questi vengono calcolati moltiplicando la superficie radiante per le ore di consumo. Le quote fisse e le spese al consumo vengono quindi sommate per ottenere l'importo totale. Il programma è attualmente dimensionato per un massimo di 20 appartamenti in modo che tutti i nominativi appaiano in un'unica videata, ma è evidente che con poche modifiche tale limite può essere aumentato notevolmente.

Come si utilizza il programma

Quando viene dato il RUN appare il seguente menu:

1 Dati costanti;
2 Dati variabili;
3 Ripartizione;
4 Stampa;
5 Fine.

I dati costanti sono: il numero degli appartamenti, i nomi dei condomini e i valori delle superfici radian-

Listato 1 - *Il programma BASIC.*

```
100 REM RIPARTIZIONE SPESE
110 REM ******************
120 REM versione
130 REM TI EXTENDED BASIC.
140 CALL CHAR(128,"00000000000000FF")
150 CALL CHAR(129,"00000000FF")
160 CALL CHAR(130,"FF")
170 CALL CHAR(131,"0101010101010101")
180 CALL CHAR(132,"5080808080808080")
190 CALL CHAR(133,"007E")
200 CALL CHAR(134,"0008103080808080")
210 DIM A$(20),T(20,6)
220 GOSUB 430
230 CALL CLEAR
240 DISPLAY AT(5,6):"RIPARTIZIONE SPESE"
250 DISPLAY AT(6,7):"DI RISCALDAMENTO"
260 CALL HCHAR(3,7,128,20)
270 CALL HCHAR(8,7,130,20)
280 CALL VCHAR(4,6,131,18)
290 CALL VCHAR(4,27,132,18)
300 CALL HCHAR(22,7,130,20)
310 DISPLAY AT(10,7)SIZE(16):"1. DATI CO
STANTI"
320 DISPLAY AT(12,7)SIZE(17):"2. DATI VA
RIABILI"
330 DISPLAY AT(14,7)SIZE(16):"3. RIPARTI
ZIONE"
340 DISPLAY AT(16,7)SIZE(10):"4. STAMPA"
350 DISPLAY AT(18,7)SIZE(7):"5. FINE"
360 DISPLAY AT(20,15)SIZE(7):"scelta?"
370 ACCEPT AT(20,23)SIZE(1)VALIDATE("123
45"):S$
380 IF S$="1" THEN 470
390 IF S$="2" THEN 680
400 IF S$="3" THEN 1010
410 IF S$="4" THEN 1690
420 IF S$="5" THEN 2270 ELSE 370
430 SUP=0 :: READ N
440 FOR J=1 TO N :: READ A$(J):: NEXT J
450 FOR J=1 TO N :: READ T(J,1):: SUP=SU
P+T(J,1):: NEXT J
460 RETURN
470 CALL CLEAR
480 PRINT "  CONTROLLO DATI COSTANTI": :
 :
490 GOSUB 1450
500 CALL HCHAR(23,3,129,28)
510 PRINT " n.    nome          sup.rad.";
 :
520 CALL HCHAR(23,3,129,28)
530 SUP=0 :: FOR J=1 TO N
540 PRINT J;TAB(7);A$(J);TAB(22);T(J,1)
550 SUP=SUP+T(J,1):: NEXT J
560 PRINT
570 CALL HCHAR(23,3,129,28)
580 PRINT "confermi? (y/n)              ";
590 ACCEPT AT(24,17)VALIDATE("YyNn"):R$
```

PERSONAL SOFTWARE

Ripartizione spese di riscaldamento

ti. Tutti sono contenuti nelle istruzioni DATA alle linee 1660, 1670 e 1680 del programma. Selezionando il numero 1 del menù è possibile verificare l'esattezza dei dati costanti ed eventualmente correggerli. Tale modifica, da programma non è permanente. L'unico modo per cambiare definitivamente i dati è di intervenire sulle linee contenenti le istruzioni DATA. L'alternativa sarebbe stata quella di creare, per i dati costanti, un file su nastro o su dischetto ma, tutto sommato, la soluzione qui adottata è la più semplice e veloce. Quando si modificano le istruzioni DATA è necessario seguire alcune norme. Il numero iniziale deve corrispondere al numero dei dati che vengono inseriti successivamente, i nomi dei condomini non devono superare i 15 caratteri altrimenti sorgeranno problemi di tabulazione sul video. I dati variabili comprendono le varie voci che concorrono alla spesa totale (fatture emesse in date diverse, bollette ENEL, ecc.) fino ad un massimo di 10 e i consumi, in ore, relativi ad ogni appartamento. Va specificata, inoltre, la percentuale spettante alle spese fisse. Selezionando il numero 3 si ha il calcolo della ripartizione ed il display dei risultati parziali o totali sul video. In ogni momento è possibile procedere o tornare alla videata precedente senza dover aspettare la fine delle operazioni per poter controllare nuovamente i risultati iniziali. Il punto quattro, infine, permette la stampa del prospetto finale nel numero di copie desiderate.
Chi è sprovvisto di stampante può tralasciare tutta la parte del programma dalla linea 1690 in poi.

Variabili del programma

CHR$(128) e **CHR$(132)** - Servono per tracciare linee orizzontali e verticali, per formare riquadri, intestazione e chiusura tabelle, ecc.
CHR$(133) - Sottolinea i dati in input per indicarne il formato.
CHR$(134) - Definisce una freccia

Seguito listato 1

```
600 IF R$="Y" OR R$="y" THEN 230
610 IF R$<>"N" AND R$<>"n" THEN 590
620 DISPLAY AT(24,1):"quale dato vuoi ca
mbiare?"
630 ACCEPT AT(24,27)SIZE(2)VALIDATE(DIGI
T):D
640 IF D<1 OR D>N THEN 630
650 DISPLAY AT(24,1):"scrivi i nuovi dat
i" :: PRINT :: PRINT D;
660 ACCEPT AT(24,7):A$(D):: ACCEPT AT(24
,27):T(D,1)
670 CALL CLEAR :: GOTO 480
680 CALL CLEAR :: TOTALE=0
690 INPUT "QUANTE VOCI CONCORRONO ALLA S
PESA TOTALE? ":V
700 PRINT :: IF V>10 THEN PRINT "SPIACEN
TE, NON PIU' DI 10" :: GOTO 680
710 PRINT
720 PRINT "SCRIVI DESCRIZIONE E IMPORTO"
:
730 FOR J=1 TO V :: PRINT
740 PRINT RPT$(CHR$(133),19);"  ";:: PRI
NT RPT$(CHR$(132),8)
750 ACCEPT AT(22,1)SIZE(18):DE$(J)
760 ACCEPT AT(22,21)SIZE(9)VALIDATE(DIGI
T):IMP(J)
770 TOTALE=TOTALE+IMP(J)
780 NEXT J
790 PRINT
800 PRINT "QUALE ALIQUOTA PERCENTUALE":"
VIENE RISERVATA ALLE SPESE"
810 INPUT "FISSE? (%) ":PERC
820 IF PERC<0 OR PERC>100 THEN PRINT "VA
LORE NON AMMESSO!" :: GOTO 790
830 PRINT
840 PRINT "confermi? (y/n) "
850 ACCEPT AT(23,18):R$
860 IF R$="Y" OR R$="y" THEN 880
870 IF R$="N" OR R$="n" THEN 680 ELSE 85
0
880 CALL CLEAR :: ORE=0 :: CONS=0
890 PRINT "CONSUMI:": : :
900 CALL HCHAR(23,3,129,28)
910 PRINT " n.   nome                ore": :
920 CALL HCHAR(23,3,129,28)
930 FOR J=1 TO N :: PRINT :: DISPLAY AT(
23,1):J :: DISPLAY AT(23,7):A$(J)
940 ACCEPT AT(23,23)SIZE(4)VALIDATE(DIGI
T):T(J,2):: ORE=ORE+T(J,2)
950 T(J,3)=T(J,1)*T(J,2):: CONS=CONS+T(J
,3)
960 NEXT J
970 PRINT :: CALL HCHAR(23,3,129,28):: D
AT$="Y"
980 DISPLAY AT(24,1):"confermi? (y/n)" :
: ACCEPT AT(24,17)SIZE(1)VALIDATE("YyNn"
):R$
```

Ripartizione spese di riscaldamento

che compare in un messaggio.
A$(20) - Contiene i nomi dei condòmini.
T(20,6) - Contiene i valori che compaiono nel prospetto finale ed è composta dai seguenti campi:
T(J,1) - Superficie radiante dell'Jesimo appartamento;
T(J,2) - Ore di consumo;
T(J,3) - Superficie radiante per ore;
T(J,4) - Quota fissa;
T(J,5) - Spesa in base al consumo;
T(J,6) - Importo spettante ad ogni condomino.
S$ - Scelta da menu.
N - Numero degli appartamenti e dei condomini.
SUP - Somma totale dei prodotti tra le superfici radianti e le ore di consumo.
R$ - Risposta (y/n); accetta caratteri maiuscoli e minuscoli.
D - Indica il numero del dato da modificare.
TOTALE - Spesa complessiva.
V - Numero delle voci che concorrono alla spesa totale.
J - Variabile di controllo per i cicli FOR NEXT.
DE$ - Descrizione delle voci che concorrono alla spesa totale.
IMP - Somma che concorre alla spesa totale.
PERC - Percentuale da addebitare come spese fisse.
ORE - Ore di consumo totale.
CONS - Somma totale dei prodotti tra le superfici radianti e le ore di consumo.
**DATI$** - Variabile flag per stabilire se sono stati introdotti i dati; per procedere al calcolo è necessario che DATI$="Y".
TOT1 - Importo da ripartire come quota fissa.
TOT2 - Importo da ripartire in base ai consumi.
K - Variabile usata nell'istruzione CALL KEY; contiene il codice ASCII del tasto premuto.
F$ - Messaggio che appare sul video per procedere o tornare alla videata precedente.
M1 e **M2** - Fattori per il calcolo di ripartizione.

Seguito listato 1

```
990 IF R$="Y" OR R$="y" THEN 230
1000 IF R$="N" OR R$="n" THEN 880 ELSE 9
80
1010 IF DATI$="Y" THEN 1030
1020 GOTO 1640
1030 GOSUB 1430 :: GOSUB 1530 :: CALL CL
EAR
1040 PRINT TAB(6);"RIPARTIZIONE SPESE":T
AB(7);"DI RISCALDAMENTO": :
1050 CALL HCHAR(23,8,129,18)
1060 PRINT :: PRINT
1070 FOR J=1 TO V :: PRINT DE$(J);TAB(19
);"L.";
1080 PRINT USING "########":IMP(J):: NEX
T J :: PRINT
1090 CALL HCHAR(23,3,129,28)
1100 PRINT "PER UN TOTALE DI";TAB(19);"L
.";
1110 PRINT USING "########":TOTALE :: PR
INT :: PRINT
1120 PRINT "DI CUI IL";PERC;"% ...";TAB(
19);"L.";
1130 PRINT USING "########":TOT1
1140 PRINT "PER SPESE FISSE": :
1150 PRINT "E LE RIMANENTI....";TAB(19);
"L.";
1160 PRINT USING "########":TOT2
1170 PRINT "DA RIPARTIRE IN BASE AI CON-
":"SUMI.": : :
1180 GOSUB 1490
1190 IF K=11 THEN 230 ELSE CALL CLEAR
1200 CALL HCHAR(23,3,129,28)
1210 PRINT "  r.   sup.    ore      ore *":
"      rad.            sup.rad.": :
1220 CALL HCHAR(23,3,129,28)
1230 FOR J=1 TO N :: PRINT USING "###":J
;
1240 PRINT ".    ";:: PRINT USING "##":T(
J,1);
1250 PRINT "    ";:: PRINT USING "####":T
(J,2);
1260 PRINT "    ";:: PRINT USING "#######
":T(J,3):: NEXT J :: PRINT
1270 CALL HCHAR(23,3,129,28):: PRINT "TO
T.  ";:: PRINT USING "###":SUP;
1280 PRINT "  ";:: PRINT USING "#####":O
RE;
1290 PRINT "  ";:: PRINT USING "########
":CONS
1300 PRINT :: CALL HCHAR(23,3,129,28)::
GOSUB 1490
1310 IF K=11 THEN 1030
1320 CALL HCHAR(23,3,129,28)
1330 PRINT "  r.  QUOTA  Q.TA AL  TOTALE
":"      FISSA  CONSUMO": :
1340 CALL HCHAR(23,3,129,28)
1350 FOR J=1 TO N :: PRINT USING "###":J
```

PERSONAL SOFTWARE

Ripartizione spese di riscaldamento

TOT3 - Somma di tutti gli importi; differisce dal totale per alcune lire dovute alle approssimazioni introdotte con la soppressione dei decimali.
DIFF - Differenza tra il totale ed il totale approssimato.
CTR - Variabile contatore usata nella routine per pareggiare il totale.
GMA - Accetta la data: giorno, mese, anno.
NCOPIE - Numero di coppie che devono essere stampate.
COPIE - Contatore per le copie stampate.

Routine utilizzate

430 - Carica in tabella i nomi e le superfici radianti.
1450 - Messaggio per premere la barra spaziatrice.
1430 - Manda il messaggio che il computer sta elaborando.
1530 - Principale routine di calcolo per la ripartizione.
1490 - Manda il messaggio per procedere o tornare alla fase precedente.

Descrizione del listato

100-210 - Definizione dei caratteri e dimensionamento delle matrici A$ e T.
220 - Rimanda alla routine per il caricamento in tabella dei nomi e del valore delle superfici radianti.
230-420 - Menu e rinvio alle varie sezioni del programma. L'istruzione ACCEPT AT accetta un dato in input da una posizione del video desiderata e permette di specificare la lunghezza del dato (SIZE) e di accettare solo particolari valori (opzione VALIDATE).
430-460 - Routine per il caricamento in tabella dei nomi e delle superfici radianti. Si trova in questa posizione che può essere giudicata anomala perchè il programma inizialmente aveva una struttura sequenziale.
470-670 - Sezione per il controllo dei dati costanti. Alla linea 590 si aspetta una conferma; se questa viene ne-

Seguito listato 1

```
1340 PRINT ".";:: PRINT USING "########":
T(J,4);
1370 PRINT USING "########":T(J,5);:: PR
INT USING "########":T(J,6)
1380 NEXT J :: PRINT
1390 CALL HCHAR(23,3,129,28):: PRINT "TO
T.";:: PRINT USING "########":TOT1;
1400 PRINT USING "########":TOT2;:: PRIN
T USING "########":TOTALE
1410 PRINT :: CALL HCHAR(23,3,129,28)
1420 GOSUB 1490 :: IF K=11 THEN CALL CLE
AR :: GOTO 1200 ELSE 230
1430 CALL CLEAR :: DISPLAY AT(13,1):"UN
MOMENTO... STO ELABORANDO"
1440 RETURN
1450 DISPLAY AT(24,11):"<space>"
1460 CALL KEY(0,K,S):: IF S=0 THEN 1460
1470 DISPLAY AT(24,1):" "
1480 RETURN
1490 F$="<ctn("&CHR$(134)&")=ind.><spac
e=avanti>"
1500 DISPLAY AT(24,1):F$
1510 CALL KEY(5,K,S):: IF S=0 THEN 1510
1520 CALL CLEAR :: RETURN
1530 TOT1=INT(PERC*TOTALE/100):: TOT2=TO
TALE-TOT1
1540 M1=TOT1/SUF :: M2=TOT2/CONS :: TOT3
=0
1550 FOR J=1 TO N :: T(J,4)=INT(T(J,1)*M
1):: T(J,5)=INT(T(J,3)*M2)
1560 T(J,6)=T(J,4)+T(J,5):: TOT3=TOT3+T(
J,6):: NEXT J
1570 DIFF=TOTALE-TOT3 :: IF DIFF=0 THEN
1620
1580 CTR=0
1590 FOR J=1 TO N :: T(J,5)=T(J,5)+1
1600 CTR=CTR+1 :: IF CTR=DIFF THEN 1620
1610 NEXT J :: IF CTR<DIFF THEN 1590
1620 FOR J=1 TO N :: T(J,6)=T(J,4)+T(J,5
):: NEXT J
1630 RETURN
1640 CALL CLEAR :: DISPLAY AT(12,1)BEEP:
"ATTENZIONE! NON SONO ANCORA STATI INTRO
DOTTI I DATI."
1650 GOSUB 1450 :: GOTO 230
1660 DATA 8
1670 DATA ROSSI G.,ALBERTI M.,VERDI E.,B
OSSANI S.,MORATTI A.,GUIDI F.,POLI P.,BE
RGAMI L.
1680 DATA 35,24,30,30,22,38,20,36
1690 IF DATI$="Y" THEN 1710
1700 GOTO 1640
1710 GOSUB 1430 :: GOSUB 1530 :: CALL CL
EAR
1720 PRINT "CONTROLLA CHE LA STAMPANTE":
:"SIA COLLEGATA ED ACCESA E": :
```

Ripartizione spese di riscaldamento

gata si passa alle linee successive per le modifiche. Specificato il numero, bisogna digitare nuovamente il nome, premere ENTER, scrivere la nuova superficie radiante e premere ancora ENTER. La modifica non è permanente ma è valida per tutta la durata dell'elaborazione attuale.

680-1000 - Sezione per l'introduzione dei dati variabili. Alla linea 770 viene calcolata la somma dei vari importi per la determinazione della spesa totale. Il programma è protetto da eventuali bizzarrie dell'utente e, ad esempio, alla linea 820 si controlla che la percentuale non sia inferiore a 0 o maggiore di 100. In altre occasioni si impedisce con l'opzione SIZE che vengano introdotti stringhe troppo lunghe o valori numerici troppo elevati.

1010-1420 - Display dei risultati sul video. Alla linea 1010 si controlla, prima dell'elaborazione, che i dati siano stati effettivamente introdotti. In caso affermativo viene mandato il messaggio di attesa per l'elaborazione che dura soltanto pochi secondi, quindi si passa alla stampa. In questa fase, come nelle successive si fa un largo uso dell'istruzione PRINT USING per il corretto incolonnamento. Si ricorda che l'istruzione TAB non può essere contenuta nella PRINT USING, pertanto vanno associate nel modo seguente: PRINT TAB();:: PRINT USING "# # # # #": variabile. Il display dell'Home Computer è troppo limitato per contenere l'intero prospetto, pertanto esso è suddiviso in tre parti: la prima per gli importi di carattere generale, la seconda per i dati necessari alla ripartizione e la terza per le quote parziali e totali. Per procedere sequenzialmente basta premere un tasto qualsiasi, per tornare alla fase precedente bisogna invece premere il tasto di funzione assieme al tasto con la freccia verso l'alto (FCTN(↑)); il codice di questa funzione è 11 ed infatti alle linee 1190, 1310 e 1420 si controlla se K ha assunto questo particolare valore.

Seguito listato 1

```
1730 PRINT "CHE LA CARTA SIA NELLA GIU-"
: :"STA POSIZIONE": : :
1740 PRINT TAB(8);"procedo? (y/n)"
1750 ACCEPT AT(23,23)SIZE(1)VALIDATE("Yy
Nn");R$
1760 IF R$="N" OR R$="n" THEN 1750
1770 PRINT " ";" " :: PRINT "SCRIVI LA D
ATA:" :: PRINT TAB(16);RPT$(CHR$(133),6)
1780 ACCEPT AT(22,16)SIZE(6)VALIDATE(DIG
IT);GMA
1790 PRINT :: PRINT :: INPUT "QUANTE COP
IE? ";NCOPIE :: CALL CLEAR
1800 DISPLAY AT(12,12):"STAMPA" :: COPIE
=0
1810 OPEN #4:"RS232"
1820 PRINT #4:CHR$(13);RPT$(" ",10);
1830 PRINT #4:CHR$(27);"E";
1840 PRINT #4:"RIPARTIZIONE SPESE DI RIS
CALDAMENTO";
1850 PRINT #4:CHR$(27);"F";CHR$(27);"G";
1860 PRINT #4:"      AL ";SEG$(STR$(GMA),
1,2);"/";SEG$(STR$(GMA),3,2);"/";SEG$(ST
R$(GMA),5,2);"."
1870 PRINT #4:" "
1880 PRINT #4:CHR$(27);CHR$(68);CHR$(29)
;CHR$(50);CHR$(53);CHR$(0);" ";
1890 FOR J=1 TO V :: PRINT #4:" ";CHR$(9
);DE$(J);CHR$(9);"L.";CHR$(9);
1900 PRINT #4,USING "########";IMP(J)::
NEXT J
1910 PRINT #4:" ";CHR$(9);RPT$("-",33)
1920 PRINT #4:" ";CHR$(9);"TOTALE";CHR$(
9);"L.";CHR$(9);
1930 PRINT #4,USING "########";TOTALE
1940 PRINT #4:" ";" ";CHR$(9);"DI CUI IL
";PERC;"% ...";CHR$(9);"L.";CHR$(9);
1950 PRINT #4,USING "########";TOT1 :: P
RINT #4:" ";CHR$(9);"PER SPESE FISSE":"
"
1960 PRINT #4:" ";CHR$(9);"E LE RESTANTI
 ...";CHR$(9);"L.";CHR$(9);TOT2
1970 PRINT #4:" ";CHR$(9);"DA RIPARTIRE
IN BASE AI CONSUMI":" "
1980 PRINT #4:CHR$(27);CHR$(68);CHR$(7);
CHR$(27);CHR$(32);CHR$(40);CHR$(48);CHR$
(57);CHR$(67);CHR$(0);
1990 PRINT #4:RPT$("-",77)
2000 PRINT #4:"  n.";CHR$(9);"Cognome e
nome";CHR$(9);"sup.";
2010 PRINT #4:CHR$(9);"   ore";CHR$(9);"
ore *";CHR$(9);" spesa";CHR$(9);"spese
al";CHR$(9);"  TOTALE"
2020 PRINT #4:" ";CHR$(9);" ";CHR$(9);"r
ad.";CHR$(9);" ";
2030 PRINT #4:CHR$(9);" sup.";CHR$(9);"
fisse";CHR$(9);"consumo"
```

Ripartizione spese di riscaldamento

Seguito listato 1

```
2040 PRINT #4:RPT$("-",77)
2050 FOR J=1 TO N
2060 PRINT #4,USING "####":J;:: PRINT #4
:".";
2070 PRINT #4:CHR$(9);A$(J);CHR$(9);
2080 PRINT #4,USING "####":T(J,1);
2090 PRINT #4:CHR$(9);:: PRINT #4,USING
"####.##":T(J,2);
2100 PRINT #4:CHR$(9);:: PRINT #4,USING
"######":T(J,3);
2110 PRINT #4:CHR$(9);:: PRINT #4,USING
"####.##":T(J,4);
2120 PRINT #4:CHR$(9);:: PRINT #4,USING
"#########":T(J,5);
2130 PRINT #4:CHR$(9);:: PRINT #4,USING
"#########":T(J,6);:: NEXT J
2140 PRINT #4:RPT$("-",77)
2150 PRINT #4:" ";CHR$(9);"TOTALI";
2160 PRINT #4:CHR$(9);:: PRINT #4,USING
"####":SUP;
2170 PRINT #4:CHR$(9);:: PRINT #4,USING
"######":ORE;
2180 PRINT #4:CHR$(9);:: PRINT #4,USING
"######":CONS;
2190 PRINT #4:CHR$(9);:: PRINT #4,USING
"####.##":TOT1;
2200 PRINT #4:CHR$(9);:: PRINT #4,USING
"#########":TOT2;
2210 PRINT #4:CHR$(9);:: PRINT #4,USING
"#########":TOTALE
2220 PRINT #4:RPT$("-",77)
2230 PRINT #4:" ":RPT$(" ",60);"L'ammini
stratore"
7240 PRINT #4:" ":RPT$(" ",60);RPT$("_",
16)
2250 PRINT #4:CHR$(12);: COPIE=COPIE+1
2260 IF COPIE<NCOPIE THEN 1820 ELSE CLOS
E #4 :: GOTO 230
2270 CALL CLEAR :: END
```

1430-1650 - Routine la maggior parte delle quali è adibita all'invio di messaggi sullo schermo (vedi: ROUTINE UTILIZZATE). La principale, situata alle linee 1530-1630, effettua il calcolo di ripartizione che corrisponde alla soluzione reiterata di proporzioni. Ad esempio, una quota fissa sta alla superficie radiante come il totale delle quote fisse sta al totale delle superfici radianti, o, usando le variabili del programma, T(J,4): T(J,1) = TOT1:SUP, da cui si ricava: T(J,4) = T(J,1) ★ TOT1/SUP e, posto M1 = TOT1/SUP, si ottiene infine T(J,4) = T(J,1) ★ M1.

1660 - L'istruzione DATA contiene il numero degli appartamenti; in questo caso otto, corrispondente anche al numero di dati successivi.

1670 - Nomi dei condomini.

1680 - Numeri che esprimono le superfici radianti.

1690-2260 - Sezione per la stampa del prospetto finale.

1690 - Controlla che siano stati inseriti i dati variabili.

1720-1760 - Messaggio per il controllo del collegamento e dell'accensione della stampante. L'intestazione del foglio è determinata dalla posizione di quest'ultimo al momento dell'accensione.

1770-1800 - Richiesta della data e del numero di copie desiderato. Per la data si scrivono di seguito giorno, mese e anno, ad esempio: 280484 (ENTER); nella stampa appariranno poi le barre separatrici: 28/04/84.

1810 - Apertura del canale per la stampa. Il numero del file, in questo caso 4, può essere scelto a piacere, tuttavia deve corrispondere a quello specificato nelle successive istruzioni PRINT.

1820 - Il carattere 13 corrisponde al ritorno carrello.

1830 - I codici 27 e 69 o ESC "E", provocano una stampa enfatizzata e danno all'intestazione un maggior risalto.

1850 - I codici ESC "F" annullano la stampa enfatizzata; si torna pertanto alla stampa normale.

1880 - I codici CHR$(27); CHR$(68); CHR$(n1); ...; CHR$(0); fissano le posizioni per una tabulazione orizzontale. È importante che la sequenza sia chiusa con il carattere con codice 0.

1890 - Il codice 9 (CHR$(9)) richiama sequenzialmente le posizioni fissate per la tabulazione orizzontale.

1900 - Si osservi il corretto formato per l'istruzione PRINT USING riferita alla stampante; al numero del file deve seguire una virgola.

1980 - Nuova definizione di posizioni per la tabulazione orizzontale. Mentre la precedente si riferiva ai dati generali, questa si riferisce ai dati particolari (vedi figura 1).

1990 - L'istruzione RPT$ ripete la stampa di uno stesso carattere o di una sequenza di caratteri. Nel caso specifico viene stampata una linea tratteggiata orizzontale.

2250 - Il codice 12 fa avanzare il foglio fino al successivo inizio pagina. L'istruzione seguente controlla il numero di copie stampato. Se questo è inferiore a quello desiderato si torna a stampare una nuova pagina altrimenti viene chiuso il file ed il programma passa al menu principale.

2270 - Chiusura del programma. ■

La rappresentazione dei numeri

153, che andremo a memorizzare dall'indirizzo 50842 in poi. Inoltre sappiamo anche che, essendo l'esponente negativo, i primi quattro bit del primo byte saranno "1001".
I secondi quattro bit vanno calcolati come segue:

1) si sottragga uno all'esponente, che consideriamo in valore assoluto;
2) si consideri la cifra delle decine del numero così ottenuto e la si sottragga a nove;
3) si trasformi il risultato di questa operazione in binario.

Nell'esempio suddetto avevamo come esponente —72. Per prima cosa sottraiamo uno al valore assoluto di quel numero, ottenendo 71. Poi eseguiamo la sottrazione 9-7 = 2. Infine trasformiamo il due in codice binario: 0010.
Il primo byte sarà dunque 10010010 = 146.

Per il secondo byte occorre un metodo analogo. Innanzitutto sappiamo già che i secondi quattro bit saranno "0000" o "1000", a seconda che il numero sia positivo, come in questo caso, o negativo.
Per i primi quattro bit si deve procedere come segue:

1) si sottragga la cifra delle unità dell'esponente al numero 10;
2) si trasformi la cifra delle unità del numero così ottenuto in codice binario.

Per il nostro numero eseguiremo: 10-2 = 8 = 1000. Il secondo byte sarà perciò 10000000 = 128. Tengo a precisare che questo complicato sistema di codifica dell'esponente deve venire effettuato solo per esponenti negativi. Uno schema di quanto detto è mostrato in figura 3.
Per risollevare un po' il morale alla fine di questo complicato discorso, e per vendicarci di questa "infernale macchinetta", fate girare il programma 1: scoprirete che non sempre due più due fa quattro, neanche per i computer!

Programma 1

```
10  A = 2 + 2
20  POKE 50842,80
30  PRINT "2 + 2 = "; A
40  END
```

Listato 1. *Con questo brevissimo programma si può "dimostrare scientificamente che due più due fa cinque"! Il trucco consiste nell'andare a modificare opportunamente uno dei byte in cui viene memorizzato il valore di A.*

Numero	Esponente	RAPPRESENTAZIONE	
		Primo byte	Secondo byte
N>0	Exp>0	0000 Decine(Exp)	Unita'(Exp) 0000
N>0	Exp<0	0000 Decine(Exp)	Unita'(Exp) 1000
N<0	Exp>0	1001 9-(Decine(Exp-1))	Unita'(10-Unita'(Exp)) 0000
N<0	Exp<0	1001 9-(Decine(Exp-1))	Unita'(10-Unita'(Exp)) 1000

Figura 3. *Schema riassuntivo della rappresentazione dell'esponente.*

TEXAS TI99/4A

Accesso diretto in un file sequenziale?!!

di Sergio Borsani

Il TI BASIC è un linguaggio molto semplice, adatto a chi vuole effettuare le prime esperienze nel campo della programmazione.
Senza comandi sofisticati, o PEEK e POKE con relative locazioni di memoria, sono veramente poche le sorprese che può riservare.
Solitamente qualche difficoltà può derivare inizialmente dall'uso della grafica oppure, e sicuramente in grado maggiore, dall'uso dei file sequenziali su nastro.
Chi non ha pensato a costituire una piccola rubrica con nomi ed indirizzi oppure un piccolo archivio per tenere in ordine la discoteca personale?
C'è chi si è spinto più in là ed ha sperato, ma solo sperato, di poter usare il TI99 senza espansioni per tenere aggiornato l'archivio di una piccola attività

Accesso diretto in un file sequenziale

commerciale!
Qual è il problema?

Nella maggior parte dei casi bisogna poter accedere ad un record in modo diretto, tramite un campo chiave, come potrebbe essere il nome di un articolo. Ma con i file sequenziali su nastro il computer deve leggere l'intero file dall'inizio, impiegando magari mezz'ora prima di individuare il record da noi richiesto!
Una soluzione consiste nel caricare l'intero file nella memoria centrale sotto forma di matrice a due dimensioni. In questo modo si può accedere direttamente ad un record, ma l'ampiezza del file deve essere drasticamente ridotta per la limitata RAM disponibile.
Come dice il proverbio, la necessità aguzza l'ingegno ... e se provassimo a leggere un file sequenziale non dall'inizio, ma esattamente da dove sappiamo debba trovarsi un particolare record?
Eresia? Prima di dare una risposta sarebbe meglio vedere all'opera il breve programma 1. Esso prevede la scrittura di record formati da cinque campi, nell'ordine: cognome, nome, indirizzo, città e numero di telefono. Naturalmente, con poche modifiche il programma può trattare altri tracciati record.
Le risposte vanno date scrivendo con caratteri maiuscoli e bisogna fare attenzione a non inserire le virgole, che servono come separatori di variabili nelle istruzioni input.
Mentre si costruisce il file, scrivendo i record ad uno ad uno sul nastro, contemporaneamente in una matrice bidimensionale prende forma un altro file, detto file-indice, contenente solo i cognomi ed il numero segnato dal tape counter prima della scrittura di ogni record. Questo numero va indicato dall'utente, il quale deve controllare costantemente il registratore in fase di scrittura.

Il file viene chiuso quando in risposta al cognome si scrive la parola FINE. Successivamente viene registrato il file-indice.

In fase di lettura si dovrà per prima cosa caricare in memoria il file-indice, dopo di che il computer potrà procedere alla ricerca di un particolare record.
Se l'utente indica un cognome presente nell'indice, riceve in risposta il numero del tape counter, in corrispondenza del quale deve essere posizionato il nastro.
Con il registratore usato per le prove tale numero deve risultare di due unità inferiore rispetto a quello indicante la posizione effettiva del record e questo perché il computer prima di leggere il nastro lo fa girare un po' a vuoto.
Comunque la sottrazione delle due unità è fatta dal programma e l'utente deve posizionare il nastro proprio al numero indicato.
Se tutto si svolge in modo corretto, il computer riesce a leggere dal file sequenziale solo un record, mostrando cognome, nome, indirizzo, città e numero di telefono del nominativo richiesto.

In caso contrario vengono letti al massimo 3 record, dopo di che, se la ricerca è stata infruttuosa, appare un messaggio di record non presente, anche se in verità, durante le prove, ciò non si è mai verificato. Qual è l'enorme vantaggio di una simile applicazione? La spiegazione è evidente; in memoria centrale non si deve tenere l'intero file, ma solo l'indice e tramite questo si può accedere direttamente ad una grande quantità di dati conservati sul nastro, proprio come avviene con un sistema a dischi. I tempi di accesso sono ragionevoli. Mentre la lettura del file sequenziale dall'inizio potrebbe comportare tempi d'attesa di mezz'ora o più, in questo modo, per la ricerca di un record, il tempo scende notevolmente e dovrebbe oscillare tra i 10 e i 20 secondi.

Accesso diretto in un file sequenziale

Dal momento che la prova è stata effettuata su non più di 10 record, si potrebbe avanzare una riserva sulla precisione del contanastro nel caso di un file di grosse proporzioni.
Il metodo è certamente in una fase sperimentale e, prima di affermare che costituisce un'autentica novità, deve essere sottoposto ad un severo collaudo, soprattutto da parte dei lettori più interessati dai quali sarà gradito un giudizio circa la sua funzionalità.

Listato 1. *Programma per la creazione di un semplice archivio.*

```
100 REM  RECORD A RICERCA
110 REM  VELOCE SU NASTRO
120 REM  ****************
130 REM  versione: TI BASIC
140 CALL CLEAR
150 DIM R$(100,2)
160 PRINT "PREMI:"::
170 PRINT "1. PER LA SCRITTURA DEL FILE"
::
180 PRINT "2. PER LA LETTURA DEL FILE"::
::
190 CALL KEY(0,K,S)
200 IF S=0 THEN 190
210 IF K=49 THEN 240
220 IF K=50 THEN 630
230 GOTO 190
240 OPEN #3:"CS1",SEQUENTIAL,OUTPUT,INTE
RNAL,FIXED
250 CALL CLEAR
260 PRINT "SCRIVI, PER PROVA, ALCUNI"
270 PRINT "RECORD."
280 PRINT "PER FINIRE SCRIVI: FINE"::
290 CTR=0
300 CTR=CTR+1
310 PRINT
320 PRINT
330 PRINT "RECORD N.";CTR
340 INPUT "cognome: ":COG$
350 IF COG$="FINE" THEN 480
360 INPUT "nome: ":NOM$
370 INPUT "indirizzo: ":IND$
380 INPUT "citta': ":CIT$
390 INPUT "tel.: ":TEL$
400 INPUT "tape counter: ":TPC$
410 PRINT
420 PRINT "      CONFERMI? (Y/N)"
430 CALL KEY(0,K,S)
440 IF S=0 THEN 430
450 IF K=89 THEN 480
460 IF K=78 THEN 300
470 GOTO 430
480 N$(CTR,1)=COG$
490 N$(CTR,2)=TPC$
500 PRINT #3:COG$,NOM$,IND$,CIT$,TEL$
510 IF COG$="FINE" THEN 530
520 GOTO 300
530 CLOSE #3
540 CALL CLEAR
550 PRINT "ORA VIENE REGISTRATO IL FILE"
560 PRINT "INDICE. ANNOTA IL NUMERO DEL"
570 PRINT "TAPE COUNTER."
580 OPEN #3:"CS1",SEQUENTIAL,OUTPUT,INTE
RNAL,FIXED
590 FOR J=1 TO CTR
600 PRINT #3:N$(J,1),N$(J,2)
610 NEXT J
620 CLOSE #3
630 CALL CLEAR
640 PRINT "ORA VIENE CARICATO IN MEMO-"
650 PRINT "RIA IL FILE INDICE."
660 OPEN #3:"CS1",SEQUENTIAL,INPUT ,INTE
RNAL,FIXED
670 CTR=0
680 CTR=CTR+1
690 INPUT #3:N$(CTR,1),N$(CTR,2)
700 IF N$(CTR,1)<>"FINE" THEN 680
710 CLOSE #3
720 CALL CLEAR
730 PRINT "ORA LEGGIAMO UN RECORD CON"
740 PRINT "IL METODO DELLA RICERCA VE-"
750 PRINT "LOCE.":"SCRIVI UN COGNOME."
760 PRINT
770 INPUT "cognome: ":REC$
780 FOR J=1 TO CTR
790 IF N$(J,1)=REC$ THEN 900
800 NEXT J
810 PRINT ::"NOMINATIVO ASSENTE!"::
820 PRINT " VUOI CONTINUARE? (Y/N)"
830 CALL KEY(0,K,S)
840 IF S=0 THEN 830
850 IF K=89 THEN 760
860 IF K=78 THEN 880
870 GOTO 830
880 CALL CLEAR
890 END
900 PRINT ::"POSIZIONA IL NASTRO AL NUME
-"
910 NUMERO=VAL(N$(J,2))-2
920 PRINT "RO";NUMERO;"DEL TAPE COUNTER.
"
930 OPEN #3:"CS1",SEQUENTIAL,INPUT ,INTE
RNAL,FIXED
940 PROVA=0
950 PROVA=PROVA+1
960 INPUT #3:COG$,NOM$,IND$,CIT$,TEL$
970 IF COG$="FINE" THEN 1090
980 IF PROVA>3 THEN 1090
990 IF COG$<>REC$ THEN 950
1000 CLOSE #3
1010 CALL CLEAR
1020 PRINT "cognome: ";COG$
1030 PRINT "nome: ";NOM$
1040 PRINT "indirizzo: ";IND$
1050 PRINT "citta': ";CIT$
1060 PRINT "tel.: ";TEL$
1070 PRINT ::::
1080 GOTO 820
1090 CLOSE #3
1100 CALL CLEAR
1110 PRINT "RECORD NON PRESENTE"::
1120 GOTO 820
```

PERSONAL SOFTWARE

Atterraggio pericoloso per TI99

Un appassionante gioco di simulazione

di *Mauro Cristuib Grizzi*

Questo programma è stato scritto per il TI 99/4A fornito di modulo SSS Extended BASIC. Consiste in una simulazione di atterraggio guidato dal controllore di volo in condizioni di totale assenza di visibilità sulla pista. Ci si dovrà basare quindi solo sulle indicazioni dello schermo radar, e su quelle che ci pervengono via radio dal velivolo circa la sua rotta, velocità, quota, carburante e distanza dalla torre di controllo. Valutati istante per istante tutti questi parametri, dovremo fornire al pilota le indicazioni sulle operazioni da effettuare: modifica dell'inclinazione degli alettoni, potenza ai motori e variazioni di rotta tramite il timone dell'aereo.

Quasi tre quarti dello schermo del televisore sono occupati dal radar con i riferimenti assoluti delle quattro direzioni cardinali, mentre nella parte inferiore sono visualizzate le variabili di volo in continuo aggiornamento, insieme a tre "spie" di controllo che indicano, con il loro colore, il range di variazione su cui sono regolati la manetta del gas, il timone e gli alettoni.

All'inizio della simulazione l'aereo si troverà in un punto all'estremità dello schermo radar, con una rotta casuale. I comandi per alettoni e potenza ai motori sono regolati in posizione neutra: alettoni orizzontali e gas zero; il velivolo pertanto rimarrà alla stessa quota diminuendo progressivamente la sua velocità. Spostando gli alettoni sulla posizione a scendere di uno o due punti, diminuirà la quota, ma aumenterà la velocità per effetto della discesa.

Viceversa, ponendo gli alettoni in posizione di salita si produrrà un recupero di quota con una conseguente diminuzione di velocità. La velocità può essere mantenuta costante (oppure aumentata o diminuita) agendo sulla regolazione del gas in combinazione con la posizione degli alettoni. La pista di atterraggio si sviluppa sulle due direttrici perpendicolari 0-180 gradi e 90-270 gradi, con una lunghezza totale di circa 16 Km. Potrete atterrare solo imboccando esattamente la pista, e a questo scopo verrà visualizzato il messaggio ON LINE quando vi troverete esattamente su una delle due direttrici della pista. Per atterrare senza schiantarsi occorre avere i seguenti valori dei parametri di volo:

- Velocità — da 120 km/h a 200 km/h;
- Quota — 0 metri;
- Rotta — 180/0 oppure 90/270.

L'aereo ha una velocità di stallo (velocità minima possibile in volo) di

Listato 1. *Il programma Atterraggio pericoloso.*

```
100 GOTO 120
110 CALL SOUND(1,300,5):: RE
TURN
120 CALL CLEAR :: CALL SCREE
N(2):: CALL CHAR(136,""):: C
ALL COLOR(14,10,10):: FOR I=
2 TO 10 :: CALL COLOR(I,2,10
):: NEXT I
130 FOR I=65 TO 88 :: CALL C
HARPAT(I,A$):: CALL CHAR(I+3
1,A$):: NEXT I :: FOR I=9 TO
 11 :: CALL COLOR(I,2,8):: N
EXT I
140 FOR I=8 TO 20 STEP 2 ::
FOR J=8 TO 22 :: CALL HCHAR(
I,J,136):: NEXT J :: GOSUB 1
10 :: FOR J=22 TO 8 STEP -1
:: CALL HCHAR(I+1,J,136)
150 NEXT J :: GOSUB 110 :: N
EXT I
160 A$="1010ATTERRAGGIO" ::
GOSUB 1110 :: A$="1111PERICO
LOSO" :: GOSUB 1110 :: A$="1
511BYHMAURO" :: GOSUB 1110
170 A$="1611CRISTUIB" :: GOS
UB 1110
180 CALL CHAR(120,""):: CALL
 COLOR(12,8,8):: FOR I=1 TO
5 STEP 2 :: FOR J=10 TO 30 :
: CALL HCHAR(I,J,120):: NEXT
 J :: GOSUB 110 :: FOR J=30
TO 10 STEP -1
190 CALL HCHAR(I+1,J,120)::
NEXT J :: GOSUB 110 :: NEXT
I :: A$="0312sdw`rxhmrsqtldm
sr" :: GOSUB 1110 :: A$="041
6bnlotsdq" :: GOSUB 1110
200 FOR C=30 TO 8 STEP -1 ::
 CALL VCHAR(1,C,32,24):: FOR
 W=1 TO 10 :: NEXT W :: NEXT
 C
210 CALL CHARSET :: CALL CLE
AR :: FOR I=48 TO 57 :: CALL
 CHARPAT(I,A$):: CALL CHAR(I
+49,A$):: NEXT I :: R1=15 ::
 G1=1 :: A5=1
220 CALL COLOR(0,16,1,2,16,1
,3,16,1,4,16,1,9,2,13,10,2,1
3,11,13,1,12,2,15,14,16,1)
230 FOR I=5 TO 8 :: CALL COL
OR(I,8,1):: NEXT I
240 CALL SCREEN(2):: CALL CH
ARPAT(96,A$,35,B$):: CALL CH
AR(136,A$,143,B$):: CALL CHA
R(112,"007F7F7F7F7F7F7F")
250 FOR I=3 TO 15 STEP 2 ::
FOR J=1 TO 17 :: CALL VCHAR(
J,I,112):: CALL VCHAR(J,33-I
,112):: NEXT J :: GOSUB 110
:: FOR J=17 TO 1 STEP -1 ::
CALL VCHAR(J,I+1,112
)
260 CALL VCHAR(J,32-I,112)::
 NEXT J :: GOSUB 110 :: NEXT
 I
270 CALL CHAR(128,"101038549
210381010381092543810100010
0844FF44081000081022FF221008"
)
280 CALL CHAR(132,"80403E302
A242A2001027C040C542454A444A
```

Atterraggio pericoloso per TI99

Seguito programma Atterraggio.

```
414047C020125222528203E4080"
)
290 CALL CHAR(48,"001C22262A
32221C"):: CALL CHAR(127,"49
C90000C90000C9")
300 CALL CHAR(137,"003098FE9
83000000000003C000000004428F
83C12698402")
310 CALL CHAR(123,"494941414
949414100FF0000990000FF41414
94941414949007F00004C00007F"
)
320 CALL CHARPAT(82,A$):: CA
LL CHAR(40,A$,113,A$):: CALL
 CHARPAT(71,A$,65,B$):: CALL
 CHAR(41,A$,114,A$,42,B$,115
,B$)
330 CALL DELSPRITE(ALL):: CA
LL MAGNIFY(1):: RANDOMIZE ::
 GOSUB 850 :: D=110 :: V=450
 :: H=3000 :: A=0 :: G=0 ::
T=R :: F=1500
340 L=INT(4*RND)+1 :: IF L=1
 THEN X=(230*RND)+8 :: Y=2 :
: R=INT(9.1*RND)*10
350 IF L=3 THEN X=8 :: Y=(13
0*RND)+8 :: R=INT(18.1*RND)*
10
360 IF L=4 THEN X=238 :: Y=(
130*RND)+8 :: R=INT(18*RND)*
10+180
370 IF L=2 THEN X=(230*RND)+
8 :: Y=130 :: R=INT(9*RND)*1
0+90
380 CALL SPRITE(#1,129,16,Y,
X)
390 P=1500 :: A1=A1+1 :: IF
R>330 OR R<30 THEN CODE=129

400 IF R>150 AND R<210 THEN
CODE=128
410 IF R>=30 AND R<=60 THEN
CODE=134
420 IF R>=210 AND R<=240 THE
N CODE=132
430 IF R>=300 AND R<=330 THE
N CODE=135
440 IF R>60 AND R<120 THEN C
ODE=130
450 IF R>240 AND R<300 THEN
CODE=131
460 IF R>=120 AND R<=150 THE
N CODE=133
470 H=H-A*.25*(V/6):: F=F-10
-ABS(G*.25)*10+A :: V=V-15+G
*5+5*A :: S=((V/3.6)*15)/400
 :: X=X+S*COS(PI/2-R/57.32):
: Y=Y+S*SIN(PI/2-R/57.32)
480 CALL PATTERN(#1,CODE)::
DQ=(ABS(X-122))^2+(ABS(Y-64)
)^2 :: D=SQR(ABS(DQ)):: IF F
<0 THEN F=0
490 A3=A3+1 :: CALL SOUND(1,
1000,0):: CALL LOCATE(#1,Y,X
):: GOSUB 990 :: IF H<1 THEN
 H=0
500 GOSUB 1020 :: IF H=0 AND
 D<=8.9 AND V<=200 THEN 740

510 IF H=0 AND V>200 THEN 83
0
520 IF (INT(RND*4)+50/H)>=5
```

120 km/h, quindi attenzione a non rallentare troppo!

Comandi sulla tastiera

Rotta

● Tasto "9": la pressione di questo tasto provoca il cambiamento del set di variazione della rotta a seconda del colore assunto dalla spia "R". Spia di colore bianco: variazioni di rotta di 30 gradi alla volta. Spia di colore verde: variazioni di rotta di 5 gradi alla volta.

● Tasto "S": diminuzione di rotta. Provoca una diminuzione del valore in gradi della rotta di 5 o di 30 gradi alla volta a seconda del colore della spia "R".

● Tasto "D": aumento di rotta. Provoca un aumento del valore in gradi della rotta di 5 o di 30 gradi a seconda del colore della spia "R".

Gas

● Tasto "0": la pressione di questo tasto provoca il cambiamento del set di variazione della manetta del gas a seconda del colore assunto dalla spia "G". Spia di colore bianco: variazione della posizione della manetta del gas di 3 unità alla volta. Spia di colore verde: variazione di 1 unità alla volta.

● Tasto ">": aumento di gas. Provoca un aumento di potenza ai motori di un numero di unità dipendente dal colore corrente della spia "G".

● Tasto "<": diminuzione di gas. Provoca una diminuzione di potenza ai motori di un numero di unità dipendente dal colore corrente della spia "G". Se il valore del gas è già 0, la pressione di questo tasto non ha alcun effeto.

Alettoni

● Tasto "=": la pressione di questo tasto provoca il cambiamento del set di variazione della posizione degli alettoni a seconda del colore assunto dalla spia "F". Spia di colore bianco: variazione della posizione degli alettoni di 3 unità alla volta. Spia di colore verde: variazione di 1 unità alla volta.

● Tasto "E": aumento di quota. Provoca una variazione della posizione degli alettoni (a salire) di una quantità dipendente dal colore corrente della spia "F".

● Tasto "X": diminuzione di quota. Provoca una variazione della posizione degli alettoni (a scendere) di una quantità dipendente dal colore corrente della spia "F".

Altri comandi

● Tasto ENTER : la pressione di questo tasto provoca una istantanea inversione di rotta di 180 gradi se ci si trova sulla direttrice 0-180 gradi, altrimenti non è attivato.

● Barra spaziatrice: la sua pressione permette l'inserimento di tutti i parametri come input statici. È consigliabile usare questo sistema fino a che non si sia acquistata una buona capacità di controllo di tutti i parametri di volo.

Tutti gli altri tasti non sono attivati, ad eccezione di Fctn (4) con funzio-

Atterraggio pericoloso per TI99

ne di break. Dovete ovviamente mantenere la figura dell'aereo all'interno del vostro schermo video, pena un messaggio di BAD VALUED IN XX e l'arresto del programma. A caso potrà comparire, tanto per vivacizzare un po' di più il gioco, un caccia nemico che cerca di colpirvi. *NOTA BENE*: il listato è stato ottenuto usando una speciale codifica per i caratteri di controllo: quando trovate in una stringa un carattere sottolineato, ciò significa che dovrete premere quel tasto insieme al tasto CONTROL. Ad esempio, "A" significa CONTROL+A.

Alcuni commenti al listato

120-200 Presentazione del gioco con definizione di caratteri speciali per avere un background color diverso da quello dello schermo. La stampa delle stringhe è ottenuta per segmenti al fine di ottenere un "effetto telescrivente".

210-320 Definizione dei caratteri speciali usati nella simulazione e dei loro colori.

330-380 Definizione dei parametri iniziali di quota, velocità, distanza, e posizione iniziale dei vari comandi; scelta casuale della posizione radar in cui compare l'aereo all'inizio del gioco, e della sua rotta.

390-460 Scelta della forma da assegnare allo SPRITE dell'aereo in funzione della sua direzione.

470-480 Aggiornamento delle variabili di volo in funzione dei comandi dati o del tempo trascorso.

490-570 Controllo sulle variabili di volo per accertare se deve essere vi-

Seguito programma Atterraggio.

```
AND D>8.9 THEN 810
530 IF INT(Y)=65 OR INT(X)=1
21 THEN DISPLAY AT(16,19)SIZ
E(7):"ON LINE" ELSE DISPLAY
AT(16,19)SIZE(7):"ppppppp"
540 IF F=0 THEN 780
550 IF V<120 THEN 800
560 IF H=0 AND D>8.9 THEN 83
0
570 IF V>1200 THEN V=1200
580 VOLTA=0 :: VEL=15 :: IF
INT(RND*1000)<15 THEN CALL S
PRITE(#2,137,10,1+RND*100,1+
RND*150,VEL,5)ELSE 600
590 GOSUB 1050 :: IF VOLTA>4
 THEN CALL DELSPRITE(#2)ELSE
 590
600 CALL KEY(0,KEY,STATUS)::
 IF STATUS=0 THEN 390
610 IF KEY=32 AND A1>2 THEN
GOSUB 960 :: A1=A1+1 :: GOTO
 390
620 GOSUB 1120
630 IF KEY=13 AND(R=0 OR R=1
80)THEN R=R+180 :: GOTO 390
640 IF R>=360 THEN R=360-R
650 IF R<0 THEN R=360+R
660 IF KEY=83 THEN R=R-R1 ::
 GOTO 390
670 IF KEY=68 THEN R=R+R1 ::
 GOTO 390
680 IF KEY=69 THEN A=A-A5 ::
 GOTO 390
690 IF KEY=88 THEN A=A+A5 ::
 GOTO 390
700 IF KEY=44 THEN G=G-G1
710 IF G<0 THEN G=0
720 IF KEY=46 THEN G=G+G1 ::
 GOTO 390
730 GOTO 390
740 IF INT(Y)=65 OR INT(X)=1
21 THEN 770 ELSE DISPLAY AT(
1,1):"PISTA NON IMBOCCATA!!!
" :: P=-1500 :: GOSUB 1100
750 CALL SOUND(1000,200,0)::
 FOR W=1 TO 1000 :: NEXT W
760 GOSUB 990 :: GOTO 840
770 DISPLAY AT(1,1):"ATTERRA
GGIO RIUSCITO!       COMPLIM
ENTI!" :: CALL SOUND(2000,50
0,0):: FOR W=1 TO 2000 :: NE
XT W :: GOTO 840
780 DISPLAY AT(1,1):"AEREO P
RECIPITATO!!!        CARBURA
NTE FINITO!!" :: P=-1500 ::
GOSUB 1100
790 CALL SOUND(1000,200,0)::
 FOR W=1 TO 2000 :: NEXT W :
: GOSUB 990 :: GOTO 840
800 DISPLAY AT(1,1):"AEREO A
NDATO IN STALLO!" :: GOSUB 1
100 :: CALL SOUND(1000,200,0
):: FOR W=1 TO 2000 :: NEXT
W :: P=-3000 :: GOSUB 990 ::
 GOTO 840
810 DISPLAY AT(1,1):"AEREO S
CHIANTATO SU OSTACOLO!!!" ::
 P=-2000 :: GOSUB 1100
820 CALL SOUND(1000,200,0)::
 FOR W=1 TO 2000 :: NEXT W :
: GOSUB 990 :: GOTO 840
830 DISPLAY AT(1,1):"AEREO S
CHIANTATO AL SUOLO!!!" :: P=
-2000 :: GOSUB 1100 :: CALL
SOUND(1000,200,0):: FOR W=1
TO 2000 :: NEXT W :: GOSUB 9
90
840 A2,A3=0 :: CALL CLEAR ::
 PRINT "PUNTEGGIO=";P :: INP
UT "VUOI RIPROVARE? ":A$ ::
IF A$="SI" OR A$="si" THEN 2
10 ELSE END
850 DISPLAY AT(1,1)SIZE(3):"
cba" :: DISPLAY AT(1,13)SIZE
(3):"bia" :: DISPLAY AT(1,26
)SIZE(3):"bfa"
860 DISPLAY AT(5,1)SIZE(3):"
cea" :: DISPLAY AT(5,26)SIZE
(3):"bca" :: DISPLAY AT(9,1)
SIZE(3):"cha"
870 DISPLAY AT(9,27)SIZE(2):
"ja" :: DISPLAY AT(13,1)SIZE
(3):"daa" :: DISPLAY AT(13,2
7)SIZE(2):"ga"
880 DISPLAY AT(17,1)SIZE(3):
"dda" :: DISPLAY AT(17,27)SI
ZE(2):"da" :: DISPLAY AT(17,
14)SIZE(1):"a"
890 CALL HCHAR(9,15,126):: C
ALL HCHAR(9,16,127):: CALL H
CHAR(9,17,124):: CALL HCHAR(
8,16,123):: CALL HCHAR(10,16
,125)
900 DISPLAY AT(18,1):"OOOOOO
OOOOOOOOOOOOOOOOOOOOOOO" :: D
ISPLAY AT(23,1):CHR$(40):: D
ISPLAY AT(23,4):CHR$(114)::
DISPLAY AT(23,7):CHR$(115)
910 DISPLAY AT(19,1)SIZE(8):
"DISTANZA" :: DISPLAY AT(19,
17)SIZE(3):"GAS"
920 DISPLAY AT(19,14)SIZE(2)
:"KM" :: DISPLAY AT(20,14)SI
ZE(2):"MT" :: DISPLAY AT(20,
26)SIZE(3):"KMH" :: DISPLAY
AT(21,14)SIZE(2):"GR"
930 DISPLAY AT(20,1)SIZE(5):
"QUOTA" :: DISPLAY AT(20,17)
SIZE(4):"VEL "
940 DISPLAY AT(21,1)SIZE(5):
"ROTTA" :: DISPLAY AT(21,17)
SIZE(5):"FLAPS"
950 DISPLAY AT(22,1)SIZE(10)
:"CARBURANTE" :: DISPLAY AT(
22,16)SIZE(1):"L" :: RETURN
960 A1=0 :: A2=A2+1 :: ON WA
RNING NEXT :: ACCEPT AT(19,2
3)SIZE(3)VALIDATE(DIGIT)BEEP
:G :: ACCEPT AT(21,23)VALIDA
TE(NUMERIC)SIZE(3)BEEP:A ::
T=R
970 ACCEPT AT(21,10)SIZE(3)V
ALIDATE(DIGIT)BEEP:R :: IF (
T=0 AND R=180)OR(T=180 AND R
=0)THEN RETURN
980 IF ABS(T-R)<=30 OR ABS(T
-R)>=330 THEN RETURN ELSE 97
0
990 REM PUNTEGGIO
1000 IF F<200 THEN P=P-100
1010 P=P-100*A2-5*A3 :: DISP
LAY AT(22,22)SIZE(2):"P=" ::
 DISPLAY AT(22,24)SIZE(6):P
:: RETURN
1020 DISPLAY AT(19,9)SIZE(4)
:INT(D):: DISPLAY AT(20,8)SI
ZE(5):INT(H):: DISPLAY AT(20
,21)SIZE(5):V :: DISPLAY AT(
21,9)SIZE(4):R
1030 DISPLAY AT(19,23)SIZE(3
):G :: DISPLAY AT(21,23)SIZE
```

Atterraggio pericoloso per TI99

sualizzato qualche messaggio.
580-590 Apparizione casuale del caccia nemico.
600-720 Analisi dei tasti premuti e conseguenti variazioni nei parametri.
730-830 Ulteriori controlli sui parametri.
850-950 Visualizzazione schermo radar e dati di volo.
960-980 Accettazione di input statici se viene premuta la barra spaziatrice.
1000-1040 Display dei dati di volo sullo schermo.
1050-1100 Apparizione del caccia nemico e controlli sulla sua traiettoria e sui suoi spari.
1120-1180 Controlli sui tasti di comando delle spie di rotta, gas ed alettoni. ■

Seguito programma Atterraggio.

```
(3):A :: IF F<200 AND F>180
THEN 1040 ELSE DISPLAY AT(22
,11)SIZE(5):INT(F):: RETURN
1040 FOR I=1 TO 5 :: DISPLAY
 AT(22,11)SIZE(5):: FOR W=1
TO 50 :: NEXT W :: DISPLAY A
T(22,11)BEEP SIZE(5):INT(F):
: FOR W=1 TO 50 :: NEXT W ::
 NEXT I :: RETURN
1050 VOLTA=VOLTA+1 :: CALL P
OSITION(#2,XP,YP):: CALL SPR
ITE(#3,138,2,XP,YP,0,50):: V
EL=-.5*VEL :: CALL MOTION(#2
,VEL,5)
1060 CALL SOUND(1,210,0):: F
OR I=1 TO 20 :: CALL COINC(#
1,#3,4,CO):: IF CO=-1 THEN 1
080
1070 NEXT I :: CALL DELSPRIT
E(#3):: RETURN
1080 CALL SOUND(1000,200,0):
: DISPLAY AT(1,1):"AEREO ABB
ATTUTO DA CACCIA   SOVIETICI
!!!" :: GOSUB 1100
1090 FOR W=1 TO 2000 :: NEXT
 W :: CALL DELSPRITE(#2,#3):
: P=0 :: GOTO 840
1100 CALL POSITION(#1,XP,YP)
:: CALL DELSPRITE(ALL):: CAL
L SPRITE(#4,139,10,XP,YP)::
CALL MAGNIFY(2):: RETURN
1110 R=VAL(SEG$(A$,1,2)):: C
=VAL(SEG$(A$,3,2)):: FOR L=5
 TO LEN(A$):: CALL HCHAR(R,C
,ASC(SEG$(A$,L,1))):: C=C+1
:: GOSUB 110 :: NEXT L :: RE
TURN
1120 IF KEY=57 AND R1=15 THE
N R1=5 :: DISPLAY AT(23,1)SI
ZE(1):CHR$(113):: GOTO 390
1130 IF KEY=57 AND R1=5 THEN
 R1=15 :: DISPLAY AT(23,1)SI
ZE(1):CHR$(40):: GOTO 390
1140 IF KEY=48 AND G1=3 THEN
 G1=1 :: DISPLAY AT(23,4)SIZ
E(1):CHR$(114):: GOTO 390
1150 IF KEY=48 AND G1=1 THEN
 G1=3 :: DISPLAY AT(23,4)SIZ
E(1):CHR$(41):: GOTO 390
1160 IF KEY=61 AND A5=3 THEN
 A5=1 :: DISPLAY AT(23,7)SIZ
E(1):CHR$(115):: GOTO 39
0
1170 IF KEY=61 AND A5=1 THEN
 A5=3 :: DISPLAY AT(23,7)SIZ
E(1):CHR$(42):: GOTO 390
1180 RETURN
```

IN MILANO - VIA MASCHERONI, 14

IL VOSTRO
"NUOVISSIMO"
COMPUTER SHOP

Libri e riviste di elettronica e informatica.

CORSI CONTINUI
TUTTO L'ANNO
CON I MIGLIORI
SOFTERISTI

IL TELEFONO È 02-437.385

POSSIEDI IL TI-99/4A E SEI ALLA RICERCA DI SOFTWARE?

Approfitta dell'offerta J. *soft!*

In seguito alle sempre più numerose richieste di programmi da parte dei possessori dell'home computer TI-99/4A, mettiamo a disposizione degli interessati un vasto assortimento a prezzi estremamente contenuti
L'offerta è valida fino ad esaurimento delle scorte.
Prima ordini, più sei sicuro di ricevere quanto desideri.

Programming Aids 1 - (Aiuto alla programmazione)
Cod. DTXSX04 - L. 14.000

Mette a disposizione gli ausilii per potenziare il linguaggio BASIC TI. Comprende le seguenti possibilità: "CATALOGARE DISCHI", "DISPLAY AT", "ACCEPT AT", "SCREEN PRINT (per la stampa su carta dello schermo)", "LOWERCASE (per la creazione di un set di lettere minuscole)", "2nd ASCII (per la costruzione e l'utilizzo di un secondo set di caratteri)", "CHARDEF (per realizzare e facilitare la definizione dei caratteri)".

Configurazione richiesta: A
Configurazione raccomandata: D + E

Blackjack e Poker
Cod. DTXMX04 - L. 25.000

Partite di carte simulate col computer, che vi consentono di puntare le somme dei vostri desideri. Possono partecipare fino a quattro giocatori

The Attack (Attacco)
Cod. DTXMX25 - L. 25.000

Vi attende il ruolo di capitano di un'astronave in una regione spaziale infestata da "spore" ed "etraterrestri" che vanno distrutti.
Manovrate l'astronave per evitare gli extraterrestri e lanciate missili per annientare il nemico.

Configurazione raccomandata: B

Blasto
Cod. DTXMX05 - L. 25.000

Un carro armato (o due, a seconda del numero dei giocatori) entrano in lizza per far saltare un campo minato, evitando nel contempo i tiri dell'avversario. Una corsa velocissima contro il cronometro allo scopo di colpire il massimo numero di mine. Attenti a non colpire le mine se siete troppo vicini, altrimenti dovrete ricominciare da capo.

Configurazione raccomandata: B

Hunt the Wumpus (Caccia la Mostro)
Cod. DTXMA 12 - L. 25.000

Una caccia emozionante in un dedalo di caverne e gallerie. Cercate la tana del Mostro evitando i pericoli in agguato lungo il percorso. Soppesate e vagliate attentamente gli indizi per completare questa missione pericolosa.

Configurazione raccomandata: B

Personal Report Generator (Generatore rapporti e archivi personali)
Cod. DTXTX01 - L. 68.000

Permette di creare, edit e stampare lettere e rapporti di routine con i dati ricavati da un archivio creato esclusivamente con i moduli "Gestione dati personali (cod. DTXGX01)" o "Dati Statistici".

Configurazione raccomandata: A o C
D + E

A - MAZE - ING
Cod. DTXMX03 - L. 25.000

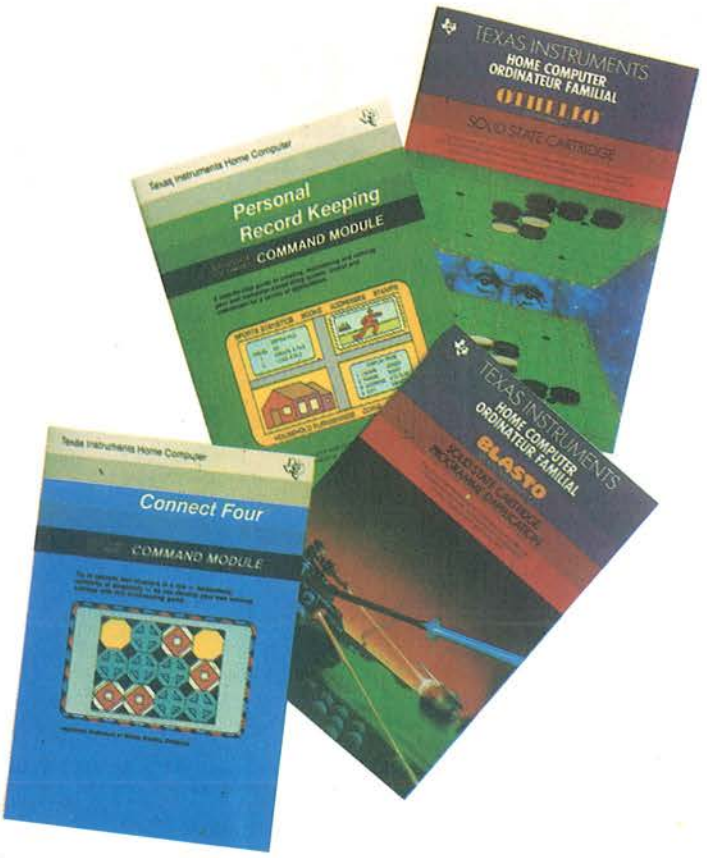
Se vi piacciono i labirinti, ecco il gioco per voi.
Opzioni diverse vi offrono una grande varietà di situazioni. Da semplici dedali alla caccia al topo. Tredici opzioni con 5.200 variazioni possibili.

Configurazione raccomandata: B

Munch Man
Cod. DTXMX16 - L. 45.000

Manovrate il Munch Man attraverso un dedalo e cercate di raggiungere uno stimolatore prima di venire divorati dai quattro Hoonos che stanno incalzando il Munch Man. Segnate punti collegando i passaggi con una catena continua oppure catturando gli astuti Hoonos mentre il Munch Man si ricarica con 10 stimolatori.

Configurazione raccomandata: B



Tombstone City: 21 secolo
Cod. DTXMX28 - L. 25.000

Vi trovate in una città fantasma del Far West nel XXI secolo, minacciata da un'orda di invasori verdi. Avanzate con la vostra pattuglia su carri coperti tipo Far West, cercando di arrestare i verdi "morgs".
Gioco velocissimo per una persona, Tombstone City mette alla prova la Vostra abilità strategica e la prontezza di riflessi.

Configurazione raccomandata: B

Market Simulation (Simulazione di mercato)
Cod. DTXIX07 - L. 14.000

Due giocatori si trovano in concorrenza d'affari. Scegliete quanta pubblicità fare, quanti pezzi volete produrre, ecc. ed attendete i risultati.
I mutamenti economici e sociali rendono estremamente realistico questo gioco istruttivo.

Configurazione richiesta: A

Othello
Cod. DTXMX20 - L. 35.000

Questo antico gioco di strategia impegna a fondo anche i giocatori più smaliziati. Un gioco da scacchiera che si impara in pochi minuti ma che richiede... una vita per dominarlo. Per strateghi di tutte le età.

Zero Zap
Cod. DTXMX34 - L. 18.000

Flipper computerizzato dal ritmo rapido, con effetti sonori e di illuminazione. Potete creare voi stessi il campo di gioco.

Connect Four (Filetto)
Cod. DTXMX08 - L. 25.000

Un impegnativo gioco di strategia verticale. I giocatori devono riuscire a collocare quattro contrassegni consecutivi in senso verticale, orizzontale o diagonale.

Yahtzee
Cod. DTXMX33 - L. 25.000

Emozionante gioco di dadi che alla strategia abbina elementi di fortuna. I giocatori accumulano punti se escono certe combinazioni.

Oldies but Goodies (Vecchi ma buoni) - Gioco II
Cod. DTXMX19 - L. 14.000

Una serie di giochi, che include Hammurabi, Hidden Paris, Peg Jump, Cerchi e croci tridimensionali e Word Safari.

Configurazione richiesta: A

Personal Record Keeping (Gestione dati personali)
Cod. DTXGX01 - L. 68.000

Permette di creare, mantenere ed utilizzare un sistema d'archivio computerizzato, utile e comodo per molte applicazioni, fra cui inventario domestico, scadenze manutenzione autovettura, cartelle cliniche, oltre a un mezzo di consultazione completa per compleanni, onomastici, anniversari e altre date importanti.

Configurazione raccomdandata: A o C D + E

Hangman (Il carnefice)
Cod. DTXIX06 - L. 25.000

Il giocatore cerca di scoprire la parola segreta e ogni volta che sbaglia, si avvicina maggiormente alla forca. Potete usare 200 parole programmate in inglese oppure crearne 60 nuove.

Configurazione raccomandata: A

Configurazione:

A - Reistratore a cassetta e cavetto di collegamento
B - Telecomandi a filo (coppia)
C - Memoria a dischi comprendente una scheda comando dischi e un comando dischi
D - Scheda interfaccia parallela o seriale RS232
E - Stampante a matrice (o altro tipo di stampante)

e inoltre 5 libri in lingua originale, estremamente utili per il tuo TI 99/4A:

Editore ARC soft - autore L. Turner:

101 Programming Tips & Tricks for the Texas Instruments TI 99/4A Home computer.
Cod. BASC001 - L. 20.000

36 Texas Instruments TI99/4A Programs for Home, School & Office
Cod. BASC002 - L. 20.000

Texas Instruments Home Computer Games Programs
Cod. BASC004 - L. 20.000

Texas Instruments Home Computer Graphics Programs
Cod. BASC003 - L. 22.000

Editore GRANADA - autore G. Marshall:

Get more from the TI99/4A
Cod. BGRC001 - L. 15.000

Ritaglia ed invia a J. soft il tagliando sotto riportato, debitamente compilato in ogni sua parte.

Spett. J. soft - Via Rosellini, 12 - 20124 Milano (MI)
Tel. 02/6888228-6880841-6880842-6880843-683797

Ordino il seguente software/libri per il TI-99/4A:

cod. .....cod. .....cod. .....
cod. .....cod. .....cod. .....
cod. .....cod. .....cod. .....

per un totale di L. ..... + L. 2.500 per contributo fisso di spese di spedizione.

Scelgo la seguente modalità di pagamento:

☐ pagherò in contrassegno al postino
☐ allego assegno (o contanti)
☐ verso l'importo sul C.C.P. n. 19445204 intestato a J. soft (allego ricevuta)

Nome .....

Cognome .....

Via ..... n. .....

CAP ..... Città ..... Prov. .....

TEXAS TI99/4A

Sottoprogrammi in TI Extended BASIC

di Sergio Borsani

Quanto è "esteso" il nostro Extended BASIC? Non è nostra intenzione di parlare diffusamente del modulo di comando SSS, che permette di ampliare il linguaggio del TI99/4A, ma piuttosto di soffermarci su una delle sue caratteristiche più notevoli e forse meno conosciute.
Se infatti si chiedesse quali sono le istruzioni più apprezzate tra quelle fornite dal modulo di estensione, sicuramente la risposta riguarderebbe i comandi grafici per la creazione ed il controllo degli sprite.
Mentre gran parte delle sue istruzioni si possono simulare anche con semplici programmi in TI BASIC (è il caso delle funzioni booleane AND e OR o di altre come DISPLAY AT, ACCEPT AT, ecc.), le funzioni grafiche per gli sprite sono invece esclusive dell'Extended BASIC. Esse inoltre risultano particolarmente agevoli per la programmazione, in quanto permettono di evitare l'uso delle POKE e la conoscenza di specifiche locazioni di memoria, come succede invece con altri computer.
Vista l'importanza e la notorietà dei comandi grafici, con molta probabilità passa inosservata un'altra caratteristica esclusiva dell'Extended-BASIC: la gestione dei sottoprogrammi. Normalmente per sottoprogrammi si intendono subroutine richiamate con un'istruzione GOSUB. I sottoprogrammi dell'Ext-BASIC sono invece qualcosa di più potente, anche se sostanzialmente analogo.

Un sottoprogramma è un programma distinto dal programma principale, non solo perché necessariamente deve essere posto al termine di esso, dopo l'istruzione END, ma anche perché le sue variabili, pur identiche a quelle del programma principale, si comportano in modo del tutto indipendente da esse. Ad esempio, se nel programma principale si è posto N=10, si può usare la variabile N anche nel sottoprogramma ed attribuire ad essa valori diversi, come 11, 12, 13, ecc. Quando si torna al programma principale, N assumerà di nuovo il valore precedente alla chiamata del sottoprogramma, cioè 10. Infatti, all'uscita di un sottoprogramma tutte le sue variabili vengono azzerate ed il programma principale riprende con le sue variabili poste al valore che avevano al momento della chiamata.
Un eventuale passaggio di valori tra il programma ed il sottoprogramma è possibile ed avviene solo sotto il controllo del programmatore.

A chi scrive non risulta che altri computer del livello medio-basso ed altre versioni di BASIC posseggano una simile capacità e al lettore attento non sfuggirà come ciò consenta una più agevole programmazione, sul tipo della "programmazione strutturata" ottenibile in Pascal.
In pratica è possibile costituire una libreria di routine da inserire, o più precisamente agganciare, ai normali programmi senza alcuna modifica e senza alcuna preoccupazione circa la compatibilità delle variabili.

A chi non è successo di "sporcare" il valore di una variabile perché inconsapevolmente aveva usato la stessa in una subroutine? Con l'uso dei sottoprogrammi ciò non succede! Quali istruzioni permettono la loro gestione?
Un sottoprogramma si apre con le istruzioni SUB "nome" (lista variabili). Essa, come è già stato detto, deve seguire il programma principale ed essere posta solo dopo un'istruzione END o una REM.
La lista delle variabili tra parentesi è il cordone ombelicale tra il programma madre ed il sottoprogramma e consente il passaggio di particolari valori nei due sensi.
Il sottoprogramma deve terminare solo con l'istruzione SUBEXIT o SUBEND, che pertanto hanno una funzione analoga al RETURN di una subroutine.
Per accedere ad un sottoprogramma si usa l'istruzione: CALL "nome" (lista variabili), che funziona come una GOSUB.
La linea delle variabili tra parentesi deve trovare una esatta corrispondenza con quella presente nell'istruzione SUB e serve per trasmettere e ricevere particolari valori. Alcuni esempi chiariranno meglio tutto il meccanismo. Battete il seguente programma.

```
100 REM SOTTOPROGRAMMA
110 REM ***************
120 CALL CLEAR :: N=9999
130 PRINT "SCRIVI UNA PAROLA" :: INPUT A
$
140 CALL REVERSE(A$)
150 PRINT " ":"  ";A$
160 PRINT :: PRINT "N =";N :: END
1000 SUB REVERSE(X$)
1010 LN=LEN(X$):: B$=""
1020 FOR N=LN TO 1 STEP -1
1030 B$=B$&SEG$(X$,N,1)
1040 NEXT N :: X$=B$
1050 SUBEND
```

La lista 140 chiama un particolare sottoprogramma, chiamato REVERSE, che accetta una stringa e la trasforma nella stringa inversa, cioè nella stessa, come risulterebbe leggendola da destra a sinistra. Nel programma principale A$ contiene la parola specificata, mentre la variabile N è stata inserita solo a titolo dimostrativo.

Sottoprogrammi in TI Extended BASIC

Il sottoprogramma riceve il contenuto di A$ nella variabile X$ e successivamente trasforma X$ nella stringa inversa. All'uscita del sottoprogramma il contenuto di X$ viene trasferito nuovamente in A$. Lo scambio dei valori avviene solo tra le variabili poste tra parentesi nella CALL e nella SUB. Le altre variabili, pur con lo stesso nome, sono indipendenti le une dalle altre.
A conferma di ciò si è usata la N come variabile di lavoro nel sottoprogramma. Nonostante ciò, tornati nel programma principale, N mantiene il suo primitivo valore, cioè 9999.
A maggior riprova della potenza e della versatilità dei sottoprogrammi c'è un'opzione che permette di trasmettere un valore dal programma al sottoprogramma senza alterare nemmeno le variabili contenute tra parentesi nella CALL. Modificate nel modo seguente il programma ed eseguitelo.

```
100 REM SOTTOPROGRAMMA
110 REM ***************
120 CALL CLEAR :: N=9999
130 PRINT "SCRIVI UNA PAROLA" :: INPUT A
$
140 CALL REVERSE((A$))
150 PRINT :: PRINT "PROGRAMMA:":"A$ = ";
A$
160 PRINT :: PRINT "N =";N :: END
1000 SUB REVERSE(X$)
1005 PRINT :: PRINT "SOTTOPROGRAMMA 1:":
"X$ = ";X$
1010 LN=LEN(X$):: B$=""
1020 FOR N=LN TO 1 STEP -1
1030 B$=B$&SEG$(X$,N,1)
1040 NEXT N :: X$=B$
1045 PRINT "SOTTOPROGRAMMA 2:":"X$ = ";X
$
1050 SUBEND
```

Questa volta il contenuto di X$ non è tornato in A$ e questo semplicemente perché A$ è stata inserita nella CALL tra un'ulteriore coppia di parentesi.
Come ultimo esempio viene riportato un sottoprogramma che, a differenza dai primi, può rivestire qualche utilità nei programmi che richiedono l'introduzione del tempo nel sistema sessagesimale.

Esso inizia alla linea 1000 del listato 1 e porta il nome convenzionale di TIME. La sintassi per richiamarlo è: CALL TIME (n° riga, n° colonna, "00:00:00.00", variabile numerica).
Dopo la CALL appare sullo schermo la stringa 00:00:00.00 nella posizione specificata dal numero di riga e di colonna (il sottoprogramma non è protetto contro quei valori che porterebbero parzialmente o totalmente la stringa fuori dallo schermo).
Un cursore, con la forma di una lineetta bianca, si posiziona all'inizio della stringa e l'utente può inserire il tempo, specificando ore, minuti e secondi, ed usare i tasti di funzione FCTN (←) e FCTN (→) per editare ciò che ha scritto.
Dopo la scrittura delle ore il cursore passa automaticamente ai minuti e successivamente ai secondi, saltando il segno ":" o il punto che separa i secondi dai centesimi.
Il sottoprogramma trasforma il tempo tutto in secondi e restituisce il valore calcolato nell'ultima variabile presente tra parentesi nell'istruzione CALL che, in questo particolare esempio, è stata posta uguale a S.
È inutile aggiungere che un simile sottoprogramma può essere vantaggiosamente agganciato a tutti i programmi di matematica e fisica che trattano il tempo.
Chi poi possiede un sistema a dischi non dovrà nemmeno trascriverlo tutte le volte, poiché si limiterà semplicemente ad usare il comando MERGE.
Quando il sottoprogramma SUBTIME è agganciato al programma principale, l'istruzione CALL TIME si comporta come qualsiasi altra istruzione implementata nel modulo di comando, con una precisa sintassi da rispettare.
Il linguaggio Ext-BASIC risulterà arricchito ed ampliato in modo personale dall'utente.

Listato 1 - *Il programma Time.*

```
100 REM  SOTTOPROGRAMMA
110 REM     TIME ENTRY
120 REM ****************
130 REM      versione
140 REM TI EXTENDED BASIC
150 CALL CLEAR :: CALL CHAR(128,"0000000
00000FFFF")
160 CALL TIME(12,10,"00:00:00.00",S)
170 PRINT "SECONDI =";S
180 END
1000 SUB TIME(R,C,B$,TEMPO)
1010 DISPLAY AT(R,C)BEEP:B$
1020 Y=(R-1)*8+1 :: X=(C+1)*8+1 :: CALL
SPRITE(#1,128,16,Y,X)
1030 LN=LEN(B$):: C1=C :: C2=C+LN-1
1040 CALL KEY(5,K,S):: IF S=0 THEN 1040
1050 IF K=8 AND C>C1 THEN C=C-1 :: W=-1
:: GOTO 1140
1060 IF K=9 AND C<C2 THEN C=C+1 :: W=1 :
: GOTO 1140
1070 IF K=32 THEN W=1 :: GOTO 1130
1080 IF K=13 THEN 1160
1090 IF K<48 OR K>57 THEN 1150
1100 CALL HCHAR(R,C+2,K):: W=1 :: IF C=C
1 THEN B$=CHR$(K)&SEG$(B$,2,LN-1):: GOTO
 1130
1110 IF C=C2 THEN B$=SEG$(B$,1,LN-1)&CHR
$(K):: GOTO 1130
1120 B$=SEG$(B$,1,C-C1)&CHR$(K)&SEG$(B$,
C-C1+2,C2-C)
1130 IF C<C2 THEN C=C+1
1140 IF C=C1+2 OR C=C1+5 OR C=C1+8 THEN
C=C+W
1150 X=(C+1)*8+1 :: CALL LOCATE(#1,Y,X):
: GOTO 1040
1160 TEMPO=VAL(SEG$(B$,1,2))*3600+VAL(SE
G$(B$,4,2))*60+VAL(SEG$(B$,7,5))
1170 CALL DELSPRITE(#1)
1180 SUBEND
```

PERSONAL SOFTWARE

Word processor per TI99/4A

Per studiare con il computer e immagazzinare tutto su nastro magnetico

di *Stefano di Tommaso*

Introduzione

Cosa significa possedere un personal computer? Molto spesso significa utilizzarlo per mille cose che altrimenti sarebbero state fatte con procedure tradizionali.
Tra queste sicuramente l'elementare operazione del prendere carta e matita e scrivere degli appunti. Ma per far ciò con il computer bisogna che in memoria vi siano delle istruzioni che ordinano al computer di ricevere gli appunti sotto forma di dati, di catalogarli e di trasferirli in memoria.
Da queste elementari considerazioni hanno avuto l'avvio i primi programmi di trattamento dei testi, cioé di word processing, con l'aggiunta però di numerose sofisticazioni in ordine alla possibilità di maneggiare i testi stessi, modificarli in tutto o in parte, ricomporli in ordine eccetera.
Presentiamo in questo numero finalmente qualcosa di simile anche per i lettori che dispongono del Texas TI99/4A, spiegandone inoltre il funzionamento ed il modo in cui è strutturato.

Cosa può fare

È un programma che vi permetterà di scrivere tutto quello che volete sul video del vostro computer sfruttandone le buone capacità di editing per spostare le parole FCTN (2)+SPACE BAR; cancellarle FCTN (1); inserirle FCTN (2) + PAROLA, scrivendo riga per riga, pagina per pagina, per poi trasferire le pagine su cassetta in maniera abbastanza efficiente.
Inoltre potrete rileggere tutto quanto scritto, oppure solo una parte,

Listato 1. *Il programma Word processor.*

```
???
1000 CALL CLEAR
1010 CALL SCREEN(5)
1020 CALL COLOR(13,3,3)
1030 FOR I=1 TO 170
1040 READ V,H
1050 CALL VCHAR(V,H,128,1)
1060 DATA 4,7,4,11,4,13,4,14,4,1
5,,16,4,17,4,19,4,20,4,21,4,22,
4,24,4,25,4,26
1070 DATA 5,7,5,11,5,13,5,17,5,1
9,5,22,5,24,5,27
1080 DATA 6,7,6,11,6,13,6,17,6,1
9,6,22,6,24,6,27
1090 DATA 7,7,7,9,7,11,7,13,7,17
,7,19,7,21,7,24,7,27,8,8,8,10,8,
13,8,14,8,15
1100 DATA 8,16,8,17,8,19,8,22
1110 DATA 8,24,8,25,8,26
1120 DATA 10,3,10,4,10,5,10,6,10
,8,10,9,10,10,10,11,10,13,10,14,
10,15,10,16,10,18,10,19,10,20,10
,21
1130 DATA 10,23,10,24,10,25,10,2
6,11,3,11,6,11,8,11,11,11,13,11,
16,11,18,11,23,12,3,12,4,12,5,12
,6
1140 DATA 12,8,12,11,12,13,12,16
,12,18,12,23,12,24,12,25,12,28,1
2,29,12,30,13,3,13,8,13,10
1150 DATA 13,13,13,16,13,18,13,2
3,14,3,14,8,14,11,14,13,14,14,14
,15,14,16,14,18,14,19,14,20,14,2
1
1160 DATA 14,23,14,24,14,25,14,2
6
1170 DATA 16,3,16,4,16,5,16,6,16
,8,16,9,16,10,16,11,16,13,16,14,
16,15,16,16,16,18,16,19,16,20
1180 DATA 16,21,17,3,17,8,17,13,
17,16,17,18,17,21,18,3,18,4,18,5
,14,6,18,8,18,9,18,10,18,11,18,1
3
1190 DATA 18,16,18,18,18,21,19,6
,19,11,19,13,19,16,19,18,19,20,2
0,3,20,4,20,5,20,6,20,8,20,9,20,
10
1200 DATA 20,11,20,13,20,14,20,1
5,20,16,20,18,20,21
1210 NEXT I
1220 CALL COLOR(13,5,5)
1230 FOR COL=1 TO 12
```

Seguito listato Word processor.

```
1240 CALL COLOR(COL,2,10)
1250 NEXT COL
1260 CALL CHAR(128,"FFFFFFFFFFFF
FFFF")
1270 GOSUB 2740
1280 PRINT "
                WORD   PROCESSOR
                                ": :
1290 PRINT "          OPZIONI:
  ": : : : :
1300 PRINT "              premi": :
:
1310 PRINT "(1)se vuoi leggere i
 files  (2)se vuoi scrivere pagi
ne  (3)se  hai finito
(4)se vuoi registrare i dati"
1320 PRINT "(5)se vuoi rileggere
  pagine(6)se vuoi aggiornare pa
gine(7)se vuoi cancellare tutto"
1330 R=0
1340 CALL KEY(0,R,ST)
1350 IF ST=0 THEN 1340
1360 IF (R<49)+(R>55)THEN 1340
1370 ON R-48 GO SUB 1940,1410,13
90,2110,2260,2450,2650
1380 GOTO 1270
1390 DISPLAY : : : : : : : : : :
 : : :"     CIAOOOOOO......": :
: : : : : : :
1400 END
1410 DIM A$(23,10)
1420 GOSUB 2740
1430 INPUT "quante pagine al mas
simo ?":N
1440 IF N>10 THEN 1430
1450 INPUT "da quale  pagina  sc
rivi ?":PP
1460 IF PP>N THEN 1450
1470 GOSUB 2740
1480 FOR C=PP TO N
1490 PRINT : : :TAB(9);STR$(C)&"
^a  pagina": : :
1500 FOR I=RI+1 TO 23
1510 INPUT A$(I,C)
1520 IF LEN(A$(I,C))>28 THEN 153
0 >LSE 1550
1530 A$(I,C)=SEG$(A$(I,C),1,28)
1540 PRINT A$(I,C)
1550 CALL KEY(0,RED,STRED)
1560 IF STRED=0 THEN 1550
```

modificare, aggiornare e cancellare tutte le pagine.
Se si dispone di una stampante basterà l'aggiunta di poche istruzioni per averne una copia su supporto cartaceo.
Il programma comunque è riferito alla configurazione di sistema più diffusa da noi, cioé: console, televisore, registratore a cassette.

Con questo tipo di applicazione è possibile sfruttare un vantaggio essenziale del Texas TI99/4A nei confronti di molti suoi concorrenti: quello di poter usufruire di una comoda tastiera di tipo professionale. Praticamente dunque sarà come battere a macchina, con in più la possibilità di cancellare gli errori, scrivere ordinatamente, trasferire le pagine scritte su cassetta in pochi minuti di registrazione e... "sigillarle" con un proprio codice segreto. Infatti bisognerà conoscere sia la parola (o le parole)-chiave, per accedere alla lettura del nastro magnetico, sia il numero del file sul quale si è trasferito il testo scritto, altrimenti il programma si ferma. Da notare è che la parola-chiave non è scritta nel listato, ma sul nastro stesso, e non si potrà mai leggerla: bisogna conoscerla.
Data la limitata estensione della memoria RAM del computer non è possibile inserire contemporaneamente in memoria più di 10 pagine dattiloscritte. Ma ciò non costituisce un reale limite all'operatività del programma poiché si possono caricare i dati sul supporto magnetico e continuare nella scrittura finché si vuole.
Su una cassetta di tipo C 60 sarà possibile inserire fino a 70 pagine dattiloscritte. Con un buon risparmio d'ingombro, dunque.

WORD PROCESSOR

OPZIONI

Premi
(1) Se vuoi leggere il file
(2) Se vuoi scrivere pagine
(3) Se vuoi finire
(4) Se vuoi registrare i dati
(5) Se vuoi rileggere pagine
(6) Se vuoi aggiornare pagine
(7) Se vuoi cancellare tutto

Figura 1. *Il menu d'apertura.*

Come funziona il programma

Il programma si apre con il titolo word processor scritto su tutta l'ampiezza del video, mentre si compone lentamente, riga per riga, per poi cancellarsi con lo scrolling orizzontale, da sinistra verso destra.
A questo punto compare il menù d'apertura, riportato nella figura 1.
Ai punti (1) e (4) ci si rivolge direttamente all'utilizzo dei file, rispettivamente, per estrarli dal nastro, e per scriverli. Non prima però di aver scritto la parola segreta ed il numero del file, separati da una virgola. Al punto (2) si va a scrivere nuove pagine, scegliendo il numero totale delle pagine da comporre, e la prima pagina dalla quale scrivere. Per esempio si può scegliere di scrivere su 8 pagine a partire però dalla 4 pagina, se le precedenti 3 erano già state scritte.
Si scrive di riga in riga (figura 2) componendo le parole fino all'esatta lunghezza di ogni riga, con 28

3ª Pagina

Caro Ernesto,
bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla

Figura 2. *La scrittura avviene di riga in riga.*

caratteri disponibili; se si va oltre questi, quando premete ENTER appare la riga come l'avete composta, ma tagliata oltre il 28° carattere. Se a questo punto ciò vi sta bene, premete ancora ENTER e passate alla riga successiva, altrimenti potete scegliere fra una serie di operazioni alternative corrispondenti ad altrettanti tasti-funzione:

- FCTN (8) (=BACK) serve per scrivere da capo quella stessa riga;
- FCTN (9) (=REDO) serve per rileggere le precedenti righe della pagina sulla quale state scrivendo, già ricomposte, per proseguire poi nella scrittura;
- FCTN (6) (=PROC'D) serve per proseguire con la pagina successiva, interrompendo a quel punto la pagina che state scrivendo non prima però di averla riletta già incolonnata, e aver premuto un tasto dopo di ciò.

Quando premete PROC'D dopo la lettura della pagina appare una serie di opzioni, per scrivere da capo, per proseguire con la pagina successiva o infine per tornare al quadro principale.

Continuando a scorrere il menu principale, vediamo che al punto (3) il programma si fermerà, al punto (7) cancellerà tutto ciò che è stato scritto, al punto (5) sarà possibile leggere una o tutte le pagine. In quest'ultimo caso si prema un tasto dopo aver letto ogni pagina. Infine al punto (6) potrete aggiornare una o più pagine, inserendo il numero della pagina e quello della riga da cambiare.

Le righe di ogni pagina sono 23 per cui per ogni numero superiore che inserite dovrete rifarlo da capo. Se volete aggiungere soltanto delle righe, basterà iniziare a scrivere dalla riga che avete inserito in avanti, fino alla fine della pagina, ma se usate il comando PROC'D allora avete finito di correggere.

È importante da ultimo ricordare che, quando stabilite il numero massimo di pagine da compilare, questo numero resterà tale anche nell'operazione di trasferimento dei dati su cassetta, e sarà inserito nel file. Così se avete scritto 4 pagine ma avevate posto un massimo di 6 pagine, il registratore scriverà e leggerà dal file sempre 6 pagine, comprese quelle vuote.

Seguito listato Word processor.

```
1570 IF RED=6 THEN 1510
1580 REM
1590 IF RED=12 THEN 1660
1600 READING=99
1610 IF RED=15 THEN 1670
1620 READING=0
1630 NEXT I
1640 IF MODYFY=99 THEN 2550
1650 GOTO 1910
1660 GOSUB 2740
1670 FOR II=1 TO 23
1680 IF READING=99 THEN 1690 ELS
E 1700
1690 IF A$(II,C)="" THEN 1710
1700 PRINT A$(II,C)
1710 NEXT II
1720 IF READING=99 THEN 1730 ELS
E 1740
1730 GOTO 1620
1740 CALL KEY(0,LETTURA,STLET)
1750 IF STLET=0 THEN 1740
1760 GOSUB 2740
1770 IF MODYFY=99 THEN 2550
1780 PRINT "          opzioni":
: :
```

```
1790 IF C>=N THEN 1810
1800 PRINT "questa era la "&STR$
(C)&"^ pagina su";:N;"pagine  a
 disposizione"
1810 PRINT "(1)per rifare ultima
  pagina(2)per fare  prossima pa
gina(3)per ritornare al quadro
   principale"
1820 RIT=0
1830 CALL KEY(0,RIT,ST)
1840 IF ST=0 THEN 1830
1850 IF (RIT>51)+(RIT<49)THEN 18
30
1860 ON RIT-48 GOTO 1490,1880,18
70
1870 RETURN
1880 IF C>=N THEN 1890 ELSE 1910

1890 PRINT "          ATTENZIONE
          NON PUOI CONTINUARE"

1900 GOTO 1810
1910 NEXT C
1920 GOTO 1760
1930 REM
1940 GOSUB 2740
1950 INPUT "chiave , numero file
```

Con un'inutile perdita di tempo.
Per evitarla potrete andare al punto (6) del menu (Aggiornamento) e reinserire il nuovo numero di pagine senza cambiare alcunché.

Come è strutturato il programma

L'immagine iniziale è stata ottenuta creando un carattere pieno (quadratino), e disponendolo numerose volte (170) sullo schermo mediante l'istruzione CALL HCHAR fino a formare l'immagine voluta.
Con le istruzioni READ ... DATA il Computer legge ogni volta le coordinate di ogni punto.
Dopodiché si giunge al menu d'apertura che divide il programma in 7 subroutine principali, alcune delle quali strutturate.
Molto spesso esse si fondono tra di loro, questo per utilizzare più volte alcune istruzioni in comune. Quando ciò avviene vengono usate delle variabili "fittizie" (Reading, Modify) che assumono valore = 99 oppure 0, creando così la possibilità di non confondere le due diverse sequenze che in quel punto si incrociano.
I testi, mentre sono scritti, vengono trasferiti nel buffer input/output tramite l'istruzione di INPUT, perciò ogni volta che scrivete una riga state attribuendo una serie di valori alla variabile di stringa A$ tramite l'istruzione di INPUT.
Ecco perché alla fine di ogni riga bisognerà premere ENTER.
Per catalogare queste variabili di stringa abbiamo usato una matrice di (23x10) punti corrispondenti a 23 righe per 10 pagine.
L'istruzione in BASIC è DIM A$ (23,10).
I "puntatori" di questa matrice a due dimensioni sono le variabili (I) e (C). Per utilizzarli abbiamo costruito due cicli nidificati (figura 3).
In questo modo è possibile passare per ogni punto della matrice e compiere le operazioni desiderate.
Inoltre, quando abbiamo affrontato il problema dell'immagazzinamento

```
FOR C = 1 TO 10
        Righe di programma
FOR I = 1 TO 23
        Righe di programma
INPUT A$ (I,C)
        Righe di programma
NEXT I
        Righe di programma
NEXT C
```

Figura 3. *I due cicli nidificati per utilizzare i puntatori di DIM A$.*

Seguito listato Word processor.

```
 ?":Q$,Q
1960 OPEN #Q:"CS1",INPUT ,INTERN
AL,FIXED 192,SEQUENTIAL
1970 INPUT #Q:N,QQ$
1980 IF QQ$=Q$ THEN 1990 ELSE 19
70
1990 FOR C=1 TO N
2000 PRINT : :"STO LEGGENDO LA "
;C;"PAGINA"
2010 FOR I=1 TO 20 STEP 5
2020 PRINT "STO LEGGENDO LA ";:I
:I+1:I+2:I+3:I+4;" RIGA"
2030 INPUT #Q:A$(I,C),A$(I+1,C),
A$(I+2,C),A$(I+3,C),A$(I+4,C)
2040 NEXT I
2050 PRINT "STO LEGGENDO LA ":"
21":" 22":" 23 RIGA "
2060 INPUT #Q:A$(21,C),A$(22,C),
A$(23,C)
2070 NEXT C
2080 CLOSE #Q
2090 RETURN
2100 REM
2110 GOSUB 2740
2120 INPUT "chiave , numero file
 ?":QQ$,Q
2130 OPEN #Q:"CS1",OUTPUT,SEQUEN
TIAL,FIXED 192,INTERNAL
2140 PRINT #Q:N,QQ$
2150 FOR C=1 TO N
2160 PRINT : :"STO SCRIVENDO LA"
;C;"PAGINA"
2170 FOR I=1 TO 20 STEP 5
2180 PRINT "STO SCRIVENDO LA":;I
:I+1:I+2:I+3:I+4;"RIGA"
2190 PRINT #Q:A$(I,C),A$(I+1,C),
A$(I+2,C),A$(I+3,C),A$(I+4,C)
2200 NEXT I
2210 PRINT "STO SCRIVENDO LA":"
21":" 22":" 23 RIGA"
2220 PRINT #Q:A$(21,C),A$(22,C),
A$(23,C)
2230 NEXT C
2240 CLOSE #Q
2250 RETURN
2260 MODYFY=0
2270 GOSUB 2740
2280 PRINT "vuoi rileggere una o
 tutte le pagine (U/T)"
2290 UT=0
2300 CALL KEY(0,UT,STUT)
2310 IF STUT=0 THEN 2300
2320 IF (UT=85)+(UT=117)THEN 279
0
```

Word processor per TI99/4A

Seguito listato Word processor.

```
2330 IF (UT=84)+(UT=116)THEN 234
0 ELSE 2300
2340 FOR C=1 TO N
2350 FOR I=1 TO 23
2360 PRINT A$(I,C)
2370 NEXT I
2380 IF MODYFY=99 THEN 2520
2390 CALL KEY(0,RR,SS)
2400 IF SS=0 THEN 2390
2410 IF (UT=84)+(UT=116)THEN 243
0
2420 RETURN
2430 NEXT C
2440 RETURN
2450 REM
2460 PRINT : : : :"   reinserisc
i il numero    di  pagine  pr
eferito       (precedente = "&ST
R$(N)&" )";
2470 INPUT N
2480 INPUT "
        quale pagina vuoi aggior
nare":C
2490 IF C>10 THEN 2480
2500 MODYFY=99
2510 GOTO 2350
2520 INPUT "da quale riga ?":RI
2530 RI=RI-1
2540 GOTO 1500
2550 RI=0
2560 GOSUB 2740
2570 PRINT "           PREMI :":
: :
2580 PRINT "(1)per aggiornare an
cora    (2)per tornare al quadro
      principale"
2590 CA=0
2600 CALL KEY(0,CA,STCA)
2610 IF STCA=0 THEN 2600 ELSE 26
20
2620 IF CA-48=1 THEN 2480
2630 MODYFY=0
2640 RETURN
2650 REM
2660 FOR C=1 TO N
2670 FOR I=1 TO 23
2680 LET A$(I,C)=""
2690 PRINT "       CANCELLAZIONE
"
2700 NEXT I
2710 NEXT C
2720 N=0
2730 RETURN
2740 CALL VCHAR(1,1,32,32*24)
2750 CALL VCHAR(1,1,128,48)
2760 CALL VCHAR(1,31,128,48)
2770 RETURN
2780 REM
2790 INPUT "quale pagina vuoi ri
leggere":C
2800 GOSUB 2350
2810 GOTO 2440
```

dei testi nei file su cassetta, abbiamo trovato utilissimo il fatto che ogni riga fosse già "squadrata" e dimensionata al massimo su 28 caratteri, perché così abbiamo potuto stabilire in anticipo la lunghezza del Record logico (28 caratteri + 1 x 5 righe), ed adattarlo su un record fisico che lo contenesse ("Fixed 192"), variando, al variare della lunghezza del testo, solo il numero delle pagine.
Infine una parola sullo scrolling orizzontale. È stato ottenuto semplicemente con l'istruzione: CALL VCHAR (1,1,32,32★24).

Conclusione

Abbiamo visto a grandi linee come funziona il programma, e, nei limiti delle capacità del Texas TI99/4A può essere comodamente applicato per usi concreti, riteniamo. Alcune precisazioni però prima che sia troppo tardi.
Il programma occupa tutta la memoria disponibile del computer, dopo aver lasciato esattamente lo spazio sufficente ad immagazzinare il testo di 10 pagine alla volta. Questo non è ovviamente modificabile, se non in piccoli dettagli, ma con il rischio di andare in **Memory full in...** e perciò perdere tutte e 10 le pagine scritte.

Variabili usate

V,H Coordinate di ogni punto della figura iniziale.
COL Variabile corrispondente al Set di colori usati.
R,RIT,LETTURA,UT,RR,CA Variabili di ritorno delle istruzioni CALL KEY usate.
N Numero massimo di pagine da scrivere.
PP Prima pagina da cui partire nella scrittura dei testi.
C Contatore delle pagine usate nel ciclo FOR-NEXT.
I Contatore delle righe di ogni pagina.
RI Prima pagina dalla quale partire nella scrittura (quando si sta aggiornando le pagine già scritte).
A$ (I,C) = Matrice di 10 x 23 punti contenenti le parole.
RED Codice ASCII dei tasti funzione utilizzati nell'operazione di scrittura.
READING, MODYFY Variabili fittizie che discriminano il percorso del programma.
Q$ Chiave per aprire il contenuto del file immagazzinato su cassetta.
Q Numero del file. ■

nome spicchi con certi attributi (colore, lampeggio, ...), e magari in una posizione determinata dello schermo (ad esempio caricando un programma composto di più parti, il nome completo venga a comporsi al centro dello schermo a mano a mano che si caricano le varie sezioni di nastro, magari colorando diversamente varie parole, e raggiungendo così nomi più lunghi dei 10 caratteri solitamente ammessi), si può utilizzare il comando SAVE con queste varianti:
a) SAVE CHR$ 16 + CHR$ n + "nome" (dove n è il codice di colore INK, 0 ÷ 7)
oppure SAVE "(modo E) (Caps SHIFT + tasto n) nome"
(naturalmente invece del carattere di controllo 16 per INK ci può essere un altro per controllo di un diverso attributo, o più in fila);
b) SAVE CHR$23 + CHR$ c + "nome" (dove c è il numero di colonna a partire da cui verrà stampato il "nome")
oppure SAVE CHR$ 23 + "(c) nome" (numero e nome di seguito, senza spazi);
c) SAVE CHR$ 22 + CHR$ r + CHR$ c + "nome" (dove r, c sono il numero di riga e di colonna della posizione da cui verrà stampato "nome").
Nei casi b) e c) naturalmente si potrà anche aggiungere il/i carattere/i di controllo per gli attributi, come in a).
Provate qualcuno di questi SAVE (badate solo che il nome va accorciato per non superare i 10 caratteri complessivi per singolo SAVE) e osservate cosa succede al LOAD!

TEXAS TI99 4/A

Ring

di Sergio Borsani

Il gioco Ring, in versione TI Extended BASIC viene presentato in questa rubrica in quanto consente di illustrare una tecnica semplice, ma efficace di animazione che si basa sulla ridefinizione continua dei caratteri.
Ormai i lettori più smaliziati conoscono le prestazioni del microprocessore grafico TMS9918A che recentemente è stato adottato da più di dieci case costruttrici di micro nello standard MSX. Altresì sono note le possibilità offerte dal modulo TI Extended BASIC per la creazione ed il controllo degli sprite, cioè degli ormai cosiddetti spiritelli o disegni mobili sullo schermo. Già, perché senza gli sprite non sarebbe possibile riprodurre la complessità, la molteplicità e la velocità dei movimenti che generalmente richiede un gioco.

Delle istruzioni grafiche dell'Extended BASIC si è già parlato diffusamente in altri articoli (vedi Gli spiritelli, *Bit* Maggio 1984); in questa sede non è necessario parlare di esse dettagliatamente. Quello che invece merita maggior risalto e che sembra caratterizzante in questo programma è l'ampia definizione e ridefinizione di caratteri grafici per consentire la rappresentazione dei pugili nelle varie posizioni e rendere il più possibile realistiche le immagini in tutte le fasi dell'incontro.
I caratteri grafici definibili dall'utente non sono molti. Essi possono andare dal numero 32 di codice ASCII al numero 143, in tutto 112. Di questi, i primi 96 sono già definiti ed il loro uso per la grafica comporta la rinuncia o, ad esempio, delle lettere minuscole o addirittura di qualsiasi testo scritto. Tenendo poi conto del fatto che gli sprite definiti su una matrice di 8x8 pixel risultano troppo piccoli e che è da preferirsi un pattern 16x16 con una CALL MAGNIFY(3), si comprende come sussistano evidenti limitazioni.
In questo gioco ogni figura è composta da tre sprite aventi le dimensioni di 16x16 pixel. Il primo sprite forma il busto, il secondo le gambe ed il terzo, di diverso colore, i calzoncini e le scarpette. In tutto quindi sono necessari 12 caratteri per ogni figura in una determinata posizione. Il movimento tuttavia richiede la creazione di più sprite per la stessa figura. Se si pretendesse di definire all'inizio del programma tutti i caratteri che in esso verranno usati, si sarebbe costretti a gravi rinunce ed a ricorrere si e no ad un paio di pose stereotipate che renderebbero tremendamente noiosa l'azione. È questo il motivo per cui si è ricorsi alla tecnica, un po' inconsueta, di ridefinire più volte nel corso del programma gli stessi caratteri per mezzo di subroutine. In questo modo è stato possibile attribuire ad ogni pugile otto posizioni per il busto e cinque per le gambe, consentendo la loro visione di fronte, di fianco, da seduti e a terra o con il braccio alzato in segno di vittoria. In tutto sono stati definiti 203 caratteri, ben oltre il numero di quelli utilizzabili contemporaneamente. La tecnica consiste nell'usare un gruppo di caratteri e, durante le pause dell'azione, ridefinire i caratteri di un altro gruppo. Nel momento in cui si usano questi ultimi si definiscono i caratteri di un altro gruppo ancora e così via. È importante che il gioco preveda delle pause naturali, come in questo caso l'intervallo tra un round e l'altro, perché la definizione dei caratteri comporta un certo tempo e non sarebbe possibile farlo durante lo svolgimento dell'azione.
I risultati sono stati così soddisfacenti che, come tocco finale, non si è voluto rinunciare nemmeno alla ragazza che porta il cartello con il numero del round successivo! Manca invece l'arbitro, ma si assicura ugualmente un comportamento ineccepibile da parte dei pugili che non trattengono e non portano mai colpi sotto la cintura.

Dopo l'apparizione del titolo il programma prosegue premendo un tasto qualsiasi. L'utente può indicare a piacimento i nomi del campione e dello sfidante e deve

Ring

specificare se usa i joystick oppure no. Un'errata risposta causerebbe un messaggio d'errore durante l'esecuzione del programma. In mancanza di joystick il controllo avviene, naturalmente, tramite la tastiera. Le due alternative conducono ad uno svolgimento dell'azione sostanzialmente analogo: tuttavia i joystick sono raccomandabili non solo in quanto il loro uso è più naturale ed immediato, ma anche perché essi producono movimenti composti. Infatti il controllo da tastiera avviene con quattro tasti (quelli con le frecce per il pugile di sinistra) più uno per tirare un colpo mentre con i joystick valgono anche le altre quattro posizioni intermedie, cioè in alto a destra, in basso a sinistra, ecc. In pratica il pugile, può avanzare e contemporaneamente assumere la guardia bassa oppure retrocedere tirando un colpo, ecc.

Per i joystick valgono le seguenti regole. Alto: guardia alta; basso: guardia bassa; destra: spostamento a destra; sinistra: spostamento a sinistra; bottone di fuoco: parte un colpo alto o basso a seconda della posizione assunta dal pugile. Per la tastiera è previsto l'uso di determinati tasti e precisamente quelli con le frecce più il tasto Q, come bottone di fuoco, per il concorrente di sinistra; i tasti I, J, K, M più il tasto P per il concorrente di destra. Nella istruzione CALL KEY sono state definite le tastiere 1 e 2 con le quali i tasti assumono determinati valori, diversi dai codici ASCII (vedi manuale a pag. 91) e che si ripetono identicamente sulle due metà della tastiera che

Listato 1. *Il programma Ring.*

```
100 REM   R  I  N  G
110 REM ****************
120 CALL CLEAR
130 DISPLAY AT(12,12):"R I N G"
140 CALL HCHAR(9,11,42,13):: CALL VCHAR(
10,11,42,5):: CALL VCHAR(10,23,42,5):: C
ALL HCHAR(15,11,42,13)
150 CALL KEY(0,K,S):: IF S=0 THEN 150
160 CALL CLEAR
170 DISPLAY AT(1,1):"SCRIVI IL NOME DEL
CAMPIONE."
180 DISPLAY AT(2,1):"CHE NE DICI DI BIG
HAMMER?"
190 ACCEPT AT(4,1)SIZE(12):CPM$
200 DISPLAY AT(6,1):"ED ORA SCRIVI IL NO
ME DELLO"
210 DISPLAY AT(7,1):"SFIDANTE." :: ACCEP
T AT(9,1):SFD$
220 DISPLAY AT(12,1):"HAI I JOYSTICK? (Y
/N)" :: ACCEPT AT(12,24)SIZE(1)VALIDATE(
"YN"):R$
230 IF R$="Y" THEN JOY=1 :: GOSUB 2120 E
LSE JOY=0
240 DISPLAY AT(20,1):"BENE. ORA PREMI UN
 TASTO PER" :: DISPLAY AT(21,1):"COMINCI
ARE."
250 CALL KEY(0,K,S):: IF S=0 THEN 250 EL
SE CALL CLEAR
260 CALL CHAR(40,"0102040810204080"):: C
ALL CHAR(41,"8040201008040201"):: CALL C
HAR(42,"FF")
270 CALL CHAR(35,"FFFFFFFFFFFFFFFF")
280 CALL CHAR(43,"0103070F1F3F7FFF"):: C
ALL CHAR(44,"80C0E0F0F8FCFEFF")
290 CALL CHAR(36,"3F7F7D5555554444"):: C
ALL CHAR(45,"FFFFFFFFFFFFFFFF")
300 CALL CHAR(37,"FFFEFCF8F0E0C080"):: C
ALL CHAR(38,"FF7F3F1F0F070301")
310 CALL COLOR(1,7,1):: CALL COLOR(2,12,
1)
320 FOR J=1 TO 6 :: CALL HCHAR(14+J,9-J,
43):: CALL HCHAR(14+J,10-J,45,14+2*J)::
CALL HCHAR(14+J,24+J,44):: NEXT J
330 FOR J=1 TO 3 :: CALL HCHAR(20+J,3,35
,28):: NEXT J
340 CALL VCHAR(17,3,45,4):: CALL VCHAR(1
2,9,45,4):: CALL VCHAR(12,24,45,4):: CAL
L VCHAR(17,30,45,4)
350 FOR J=1 TO 3 :: CALL HCHAR(11+J,10,4
2,14):: NEXT J
360 FOR J=1 TO 3 :: FOR K=1 TO 5 :: CALL
 HCHAR(16+J-K,3+K,40):: CALL HCHAR(10+J+
K,24+K,41):: NEXT K :: NEXT J
370 GOSUB 1440 :: CALL SPRITE(#10,96,2,1
12,80,#11,100,2,128,80,#9,104,8,128,80)
380 CALL MAGNIFY(3)
390 CALL SPRITE(#20,96,2,112,160,#21,100
,2,128,160,#19,104,9,128,160)
400 DISPLAY AT(1,1):"    CAMPIONATO DEL
MONDO    "
410 DISPLAY AT(3,1):"CAMPIONE:" :: DISPL
AY AT(4,1):CPM$
420 DISPLAY AT(3,15):"SFIDANTE:" :: DISP
LAY AT(4,15):SFD$
430 DISPLAY AT(5,1):"ALT.: 185 CM  ALT.:
 182 CM"
440 DISPLAY AT(6,1):"PESO: 105 KG  PESO:
 102 KG"
450 DISPLAY AT(7,1):"VINTE: 65     VINTE
: 53"
460 DISPLAY AT(8,1):"PERSE: 2      PERSE
: 4"
470 DISPLAY AT(9,1):"PARI:  1      PARI:
 3"
480 FOR TIME=1 TO 2000 :: NEXT TIME
490 CALL HCHAR(1,1,32,288):: CALL COLOR(
1,5,1):: CALL SCREEN(11):: CALL COLOR(2,
12,1)
500 FOR J=1 TO 12 :: CALL HCHAR(J,1,35,1
2-J):: CALL HCHAR(J,13-J,37)
510 CALL HCHAR(J,19+J,38):: CALL HCHAR(J
,20+J,35,12-J):: NEXT J
520 C(1)=80 :: C(2)=160 :: GOSUB 1480
530 CALL SPRITE(#9,128,8,128,C(1)-4,#10,
120,2,112,C(1),#11,124,2,128,C(1)-4)
540 CALL SPRITE(#19,116,9,128,C(2)+4,#20
```

Seguito listato Ring.

```
,108,2,112,C(2),#21,112,2,128,C(2)+4)
550 GOSUB 1550 :: CALL SOUND(1000,-1,1)
560 TT=0 :: P(1),P(2)=0 :: F(1),F(2)=1 :
: ROUND=1 :: PG=2 :: W=-1
570 PG=PG+W
580 IF JOY=1 THEN 660
590 CALL KEY(PG,K,S):: IF S=0 THEN CALL
PATTERN(#PG*10,132-PG*12):: F(PG)=1 :: G
OTO 900
600 IF K=5 THEN X=0 :: Y=4 :: GOTO 670
610 IF K=0 THEN X=0 :: Y=-4 :: GOTO 670
620 IF K=2 THEN X=-4 :: Y=0 :: GOTO 670
630 IF K=3 THEN X=4 :: Y=0 :: GOTO 670
640 IF K=18 OR K=11 THEN 810
650 GOTO 900
660 CALL JOYST(PG,X,Y)
670 IF X=0 AND Y=0 THEN 800
680 IF X=0 THEN 740
690 CALL DISTANCE(#10,#20,DD):: IF DD<60
 AND SGN(X)<>W THEN 740
700 IF C(PG)+X>200 THEN 740
710 IF C(PG)+X>50 THEN C(PG)=C(PG)+X ::
CALL LOCATE(#PG*10,112,C(PG),#PG*10+1,12
8,C(PG)+4*W,#PG*10-1,128,C(PG)+4*W)
720 IF PG=1 THEN Z1=76 :: Z2=80 ELSE Z1=
64 :: Z2=68
730 CALL PATTERN(#PG*10+1,Z1,#PG*10-1,Z2
)
740 IF Y=0 THEN 800
750 IF Y=4 THEN 780
760 F(PG)=2 :: IF PG=1 THEN Z=84 ELSE Z=
136
770 CALL PATTERN(#PG*10,Z):: GOTO 800
780 F(PG)=1 :: IF PG=1 THEN Z=120 ELSE Z
=108
790 CALL PATTERN(#PG*10,Z)
800 CALL KEY(PG,K,S):: IF S=0 THEN 900
810 IF PG=1 THEN 840
820 IF F(PG)=1 THEN Z=132 ELSE Z=140
830 GOTO 850
840 IF F(PG)=1 THEN Z=72 ELSE Z=88
850 CALL PATTERN(#PG*10,Z):: KK(PG)=F(PG
)
860 CALL DISTANCE(#10,#20,DD):: IF DD>60
 THEN 900
870 CALL SOUND(100,-7,10)
880 IF KK(PG)<>F(PG-W)THEN P(PG)=P(PG)+1
890 IF P(PG)>100 THEN 990
900 TT=TT+1 :: IF TT>200 THEN CALL SOUND
(1000,-1,1):: GOTO 1230
910 NN=INT(RND*2)+1
920 IF PG=1 THEN 950
930 IF NN=1 THEN Z1=112 :: Z2=116 :: GOT
O 970
940 Z1=64 :: Z2=68 :: GOTO 970
950 IF NN=1 THEN Z1=124 :: Z2=128 :: GOT
O 970
960 Z1=76 :: Z2=80
970 CALL PATTERN(#PG*10+1,Z1,#PG*10-1,Z2
)
980 W=-W :: GOTO 570
990 IF PG=1 THEN 1040
1000 CALL PATTERN(#9,128,#10,120,#11,124
):: GOSUB 1700
1010 CALL LOCATE(#9,128,C(1)-16,#10,128,
C(1),#11,128,C(1)-16)
1020 CALL PATTERN(#10,96,#11,100,#9,104)
:: CALL LOCATE(#20,96,160,#21,112,164,#1
9,112,164)
1030 CALL PATTERN(#20,108):: GOTO 1070
1040 CALL PATTERN(#19,116,#20,108,#21,11
2):: GOSUB 1660
1050 CALL LOCATE(#19,128,C(2)+16,#20,128
,C(2),#21,128,C(2)+16):: CALL LOCATE(#10
,96,80,#11,112,76,#9,112,76)
1060 CALL PATTERN(#20,96,#21,100,#19,104
):: CALL PATTERN(#10,120)
1070 KD=KD+1 :: P(1)=P(1)-10 :: P(2)=P(2
)-10
1080 FOR TIME=1 TO 1000 :: NEXT TIME
1090 C(1)=80 :: C(2)=160 :: CALL DELSPRI
TE(ALL)
1100 CALL SPRITE(#9,128,8,128,C(1)-4,#10
,120,2,112,C(1),#11,124,2,128,C(1)-4)
1110 CALL SPRITE(#19,116,9,128,C(2)+4,#2
0,108,2,112,C(2),#21,112,2,128,C(2)+4)
1120 IF FLAG=1 THEN 1140
1130 IF KD<5 THEN 900
1140 GOSUB 1440
1150 CALL PATTERN(#10,96,#11,100,#9,104)
:: CALL LOCATE(#11,128,80,#9,128,80)
1160 CALL PATTERN(#20,96,#21,100,#19,104
):: CALL LOCATE(#21,128,160,#19,128,160)
1170 GOSUB 2080 :: CALL PATTERN(#PG*10,6
4,#PG*10+1,68)
1180 CALL HCHAR(14,PG*10+2,62)
1190 IF PARI<>1 THEN 1210
1200 CALL PATTERN(#20,64,#21,68):: CALL
HCHAR(14,22,62)
1210 FOR TIME=1 TO 3000 :: NEXT TIME
1220 CALL DELSPRITE(ALL):: CALL CLEAR ::
 END
1230 CALL PATTERN(#9,128,#10,120,#11,124
):: CALL PATTERN(#19,116,#20,108,#21,112
)
1240 ROUND=ROUND+1 :: IF ROUND>12 THEN 1
410
1250 GOSUB 1740 :: CALL HCHAR(19,5,36)::
 CALL HCHAR(19,28,36)
1260 CALL LOCATE(#10,124,34,#11,140,34,#
9,140,34):: CALL LOCATE(#20,124,208,#21,
140,208,#19,140,208)
1270 CALL PATTERN(#10,132,#11,136,#9,140
):: CALL PATTERN(#20,84,#21,88,#19,92)
1280 FOR TIME=1 TO 2000 :: NEXT TIME
1290 GOSUB 1810 :: ON ROUND-1 GOSUB 1860
,1880,1900,1920,1940,1960,1980,2000,2020
,2040,2060
1300 CALL SPRITE(#25,64,2,90,180,#26,68,
2,106,180,#24,76,14,98,180,#27,80,7,74,1
80)
1310 G=2 :: CALL MOTION(#25,0,-3,#26,0,-
3,#24,0,-3,#27,0,-3)
1320 FOR TIME=1 TO 50 :: NEXT TIME :: CH
```

così resta divisa in due parti distinte ma con analoga funzione.
Dopo le presentazioni, all'inizio del combattimento, i concorrenti non devono far altro che avvicinarsi e cominciare a darsele di santa ragione premendo i bottoni di fuoco; i punti vengono assegnati in modo imparziale dal programma con un criterio fin troppo semplice: ottiene un punto un colpo alto che trova l'avversario in posizione bassa o viceversa un colpo alla figura che trova l'avversario in guardia alta. A proposito, non dimenticate di alzare il volume del televisore perché non mancano gli effetti sonori e attenti a non maltrattare troppo le levette dei joystick che non paiono particolarmente robuste!

Descrizione del listato

100-150 - Titolo.
160-250 - Richiesta dei nomi dei pugili e dell'uso dei joystick.
260-310 - Definizione dei caratteri e dei colori per la scenografia.
320-480 - Stampa dei preliminari con la presentazione dei pugili.
490-560 - Inizio del combattimento.
570-980 - Ciclo per il controllo del movimento.
590-650 - Comandi per il movimento tramite la tastiera.
660 - Istruzione per interpretare i segnali del joystick.

La J.*soft*
CERCA
per la nuova rivista Super Apple
un redattore da assumere,
con ampia e provata
conoscenza dei sistemi Apple.
Si richiede la buona
conoscenza della lingua inglese,
residenza Milano/dintorni.
Inviare curriculum completo a
J.*soft* - Via Rosellini, 12
20124 MILANO

Seguito listato Ring.

```
=70+G :: CALL PATTERN(#26,CH)
1330 CALL POSITION(#25,SY,SX):: IF SX>70
 THEN G=-G :: GOTO 1320
1340 CALL MOTION(#25,0,0,#26,0,0,#24,0,0
,#27,0,0)
1350 FOR TIME=1 TO 1000 :: NEXT TIME ::
CALL DELSPRITE(#27,#24,#25,#26)
1360 GOSUB 1480 :: CALL PATTERN(#10,120,
#11,124,#9,128):: CALL PATTERN(#20,108,#
21,112,#19,116)
1370 CALL LOCATE(#10,112,80,#11,128,76,#
9,128,76):: CALL LOCATE(#20,112,160,#21,
128,164,#19,128,164)
1380 CALL HCHAR(19,5,45):: CALL HCHAR(19
,28,45):: CALL SOUND(1000,-1,1)
1390 CALL SOUND(1000,-1,1)
1400 GOSUB 1550 :: TT=0 :: W=-W :: C(1)=
80 :: C(2)=160 :: GOTO 570
1410 IF P(1)=P(2)THEN PG=1 :: PARI=1 ::
FLAG=1 :: GOTO 1090
1420 FLAG=1 :: IF P(1)>P(2)THEN PG=1 ::
GOTO 1090
1430 PG=2 :: GOTO 1090
1440 CALL CHAR(96,"01030303013D7F7F7F7F6
F6F6747477780C0C0C080BCFEFEFEFEF6F6E6E2E
2EE")
1450 CALL CHAR(100,"F7D7C777060606060606
060606060E3EEFEBE3EE60606060606060606060
707C")
1460 CALL CHAR(104,"07070707070700000000
000606060E3EE0E0E0E0E0E00000000000606060
707C")
1470 RETURN
1480 CALL CHAR(108,"000000000307070703"&
RPT$("0",16)&"183C3C3C9C8E9FFFFF7F1F1E1E
1E1E")
1490 CALL CHAR(112,RPT$("01",16)&"F0F0F0
F0F0F0F0F0B0988C8C88888888")
1500 CALL CHAR(116,"01010101010100000000
00000101030FB0B0B0B0B0B00000000000008C8C
8E98")
1510 CALL CHAR(120,"00183C3C3C3971F9FFFF
FEF87878787800000000C0E0E0E0C0"&RPT$("0"
,14))
1520 CALL CHAR(124,"0F0F0F0F0F0F0F0F0D19
313111111111"&RPT$("80",16))
1530 CALL CHAR(128,"0D0D0D0D0D0D00000000
000031317119C0C0C0C0C0C00000000000008080
C0F0")
1540 RETURN
1550 CALL CHAR(132,"0000000060F0FF7F"&RP
T$("0",16)&"0C1E1E1E0C3EFFFF3F3F3F1F1E1E
1E1E")
1560 CALL CHAR(136,"00000000030707071B3C
3C1C060301000000000000808 0F0FCFEFEFFFFDF
8F0F")
1570 CALL CHAR(140,RPT$("0",20)&"317F7F3
0000000307878783C1E3F7FFFFFBF1E1E1E1E")
1580 CALL CHAR(64,"0101030307070F0E1C181
```

Seguito listato Ring.

```
818181818181F0F0F0F8F8F8383818080C0C0C0C0C0C0
C0C")
1590 CALL CHAR(68,"0101030306060000000000
000181838F87070707878F8F800000000000000C0C0
E0E")
1600 CALL CHAR(72,"30787878307CFFFFFCFCF
CF878787878000000000060FFFFE"&RPT$("0",16
))
1610 CALL CHAR(76,"0F0F0F1F1F1F1C1C18303
0303030303080808C0C0E0E0F0703818181818181
818")
1620 CALL CHAR(80,"0E0E0E1E1F1F000000000
000303070708080C0C0606000000000000018181
C1F")
1630 CALL CHAR(84,"000000000001010F3F7F7
FFFFFFBF1F000000000C0E0E0E0D83C3C3860C08
000")
1640 CALL CHAR(88,"000C1E1E1E3C70FCFEFFF
FFD78787878"&RPT$("0",20)&"8CFEFE0C0000"
)
1650 RETURN
1660 CALL CHAR(96,"000000000000000307070
70F1E1E1F0F000000001F3F7FFFFFFCB0B03030F
0F0")
1670 CALL CHAR(100,"000000008F0F8FCFCFC
FCFCFCE3FFFC"&RPT$("0",32))
1680 CALL CHAR(104,"000000000000083CDCDC
DCDCDCCO"&RPT$("0",28)&"60F0F808")
1690 RETURN
1700 CALL CHAR(96,"00000000F8FCFEFFFF3F0
D0D0C0C0F0F000000000000000C0E0E0E0F07878F
8F0")
1710 CALL CHAR(100,RPT$("0",24)&"060F1F1
000000000010F1F3F3F3F3F3FC7FF3F")
1720 CALL CHAR(104,RPT$("0",24)&"060F1F1
0000000000000103C3B3B3B3B3B030000")
1730 RETURN
1740 CALL CHAR(84,RPT$("0",26)&"030F1E00
000000307878783038 7C7EFEFE3E3E")
1750 CALL CHAR(88,"1C0F0F0F0C0C0C0C00000
0000000000003EFEFEFE"&RPT$("0",24))
1760 CALL CHAR(92,"0000000000000C0C1C7C0
000000000000003A760E7E7C"&RPT$("0",22))
1770 CALL CHAR(132,"000000000C1E1E1E0C1C
3E7E7F7F7C7C"&RPT$("0",26)&"C0F078")
1780 CALL CHAR(136,"7C7F7F7F"&RPT$("0",2
4)&"38F0F0F030303030383E0000000000")
1790 CALL CHAR(140,"5C6E707E3E"&RPT$("0"
,34)&"3030383E000000000000")
1800 RETURN
1810 CALL CHAR(64,"10180804040203010103O
70703010101020204E4E8F870F8D8908080808080C
0E0")
1820 CALL CHAR(68,"0101030307060C0C0C080
808080C1424E0E0C0C0C0C0C0C0603030202020305
090")
1830 CALL CHAR(72,"0101010303030707070303O
10101010304E0E0E0C0C0808080808080C0C03028A
0A0")
1840 CALL CHAR(76,"0000080C06030303"&RPT
$("0",26)&"80C0E060200000000000000")
1850 RETURN
1860 CALL CHAR(80,"FFFFF8F0E3E3FFFFFCF1E
3E3E0E0FFFFFFFF1F0FC7C7C70F1F7FFFFF0707F
FFF")
1870 RETURN
1880 CALL CHAR(80,"FFFFF8F0E3E3FFFFFFFFE
3E3F0F8FFFFFFFF1F0FC7C7C70F0FC7C7C70F1FF
FFF")
1890 RETURN
1900 CALL CHAR(80,"FFFFFFFFFEFCF9F3E7E0E
0FFFFFFFFFFFFFF8F0F0F8F8F8F8F07078F8F07F
FFF")
1910 RETURN
1920 CALL CHAR(80,"FFFFE0E0E7E7E0E0FFFFE
7E7F0F8FFFFFFFF0707FFFF1F0FC7C7C7C70F1FF
FFF")
1930 RETURN
1940 CALL CHAR(80,"FFFFF8F0E3E3E0E0E3E3E
3E3F0F8FFFFFFFF1F0FC7FF1F0FC7C7C7C70F1FF
FFF")
1950 RETURN
1960 CALL CHAR(80,"FFFFE0E0E7FFFFFEFCF8F
8F1F1F1FFFFFFFF0707C78F1F3F7FFFFFFFFFFFF
FFF")
1970 RETURN
1980 CALL CHAR(80,"FFFFF8F0E3E3E3F0F0E3E
3E3F0F8FFFFFFFF1F0FC7C7C70F0FC7C7C70F1FF
FFF")
1990 RETURN
2000 CALL CHAR(80,"FFFFF8F0E3E3E3E3F0F8F
FE3F0F8FFFFFFFF1F0FC7C7C7C70707C7C70F1FF
FFF")
2010 RETURN
2020 CALL CHAR(80,"FFFFF3E3C3F3F3F3F3F3F
3F3F3E1FFFFFFFF870333333333333333330387F
FFF")
2030 RETURN
```

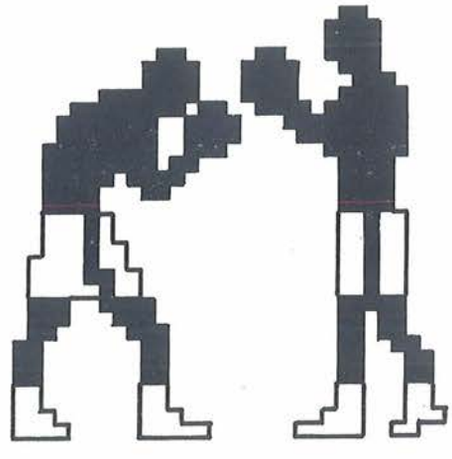


Figura 1. *Per ottenere un'azione più ricca e realistica sono stati definiti più volte, nel corso del programma, gli stessi caratteri per un numero complessivo di 203 caratteri, ben superiore quindi a quelli normalmente disponibili.*

670-790 - Controllo della posizione dei pugili sul ring, della loro distanza reciproca e della posa assunta.
800 - Controllo dei "bottoni di fuoco" dei joystick o dei tasti programmati per far partire i colpi ai pugili.
810-870 - Controllo della forma assunta dai pugili nell'atto di colpire l'avversario. Se il colpo è andato a segno, la linea 870 produce un rumore.
880-890 - Punteggio assegnato dal programma ai pugili in base ai colpi andati a segno. Se un contendente subisce più di 80 colpi subisce un atterramento.
900 - Controllo del tempo per determinare la durata dei round.

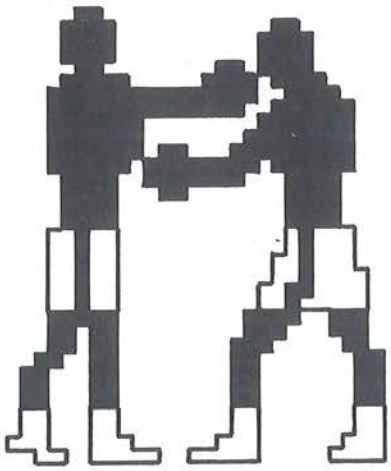


Figura 2. *Il controllo dei movimenti avviene o tramite tastiera o meglio con i joystick. Manca l'arbitro, tuttavia i pugili si comportano in modo molto corretto, non trattengono né colpiscono sotto la cintura.*

Seguito listato Ring.

```
2040 CALL CHAR(80,"FFFFF3E3C3F3F3F3F3F3F
3F3F3E1FFFFFFFFCF8F0FCFCFCFCFCFCFCFCF87F
FFF")
2050 RETURN
2060 CALL CHAR(80,"FFFFF3E3C3F3F3F3F3F3F
3F3F3E1FFFFFFFF870333F3F3E7CF9F3F3F0303F
FFF")
2070 RETURN
2080 CALL CHAR(64,"01030303013D7F7F7F7F6
F6F674747478CCCCCDC9CBCF8F8F0F0F0F0E0E0E
0E0")
2090 CALL CHAR(68,"F7D7C7770606060606060
60606060E3EE0E0E0E0606060606060606060607
07C")
2100 CALL CHAR(62,"1C1C1C0808080808")
2110 RETURN
2120 DISPLAY AT(14,1):"JOYSTICK 1 PUGILE
 DI DESTRA"
2130 DISPLAY AT(16,1):"JOYSTICK 2 PUGILE
 A SINISTRA"
2140 DISPLAY AT(18,1):"RILASCIA IL TASTO
 ALPHA LOCK"
2150 RETURN
```

910-970 - Istruzioni per produrre un certo movimento nei pugili anche in assenza di istruzioni specifiche.
980 - Assegnazione del controllo alternativamente ai due pugili e rinvio all'inizio del ciclo.
990-1130 - Routine per l'atterramento di un pugile. La linea 1070 modifica il punteggio per consentire la ripresa dell'incontro ma dopo 5 atterramenti la linea 1130 fa terminare l'incontro.
1140-1220 - Routine per il termine dell'incontro. Si alza il braccio del vincitore.
1230-1400 - Routine per l'intervallo tra un round e l'altro. I pugili si siedono e compare la ragazza con il numero del round successivo.
1410-1430 - Controllo del punteggio al termine dei 12 round.
1440-1470 - Subroutine per la definizione dei caratteri per rappresentare i pugili di fronte.
1480-1540 - Definizione dei caratteri per rappresentare i pugili nella posizione base.
1550-1650 - Definizione dei caratteri per rappresentare i pugili nelle varie pose assunte durante il combattimento.
1660-1690 - Definizione dei caratteri per il pugile di destra al tappeto.
1700-1730 - Definizione dei caratteri per il pugile di sinistra al tappeto.
1740-1800 - Subroutine per definire i caratteri che rappresentano i pugili seduti.
1810-1850 - Subroutine per la definizione dei caratteri che rappresentano la ragazza che porta il cartello con il numero del round.
1860-2070 - Varie subroutine, richiamate dalla linea 1280, per definire ad ogni round i caratteri per il cartello con il numero corrispondente.
2080-2110 - Definizione dei caratteri per rappresentare i pugili con il braccio alzato in segno di vittoria.
2120-2150 - Messaggio sull'uso dei joystick.

Principali variabili

CPM$, SFD$ - Nome del campione e dello sfidante.
JOY - Variabile flag per indicare l'uso dei joystick.
C(1), C(2) - Contengono i numeri di colonna, in pixel, della posizione degli sprite rappresentanti i pugili.
TT - Variabile per il controllo del tempo. Viene incrementata di un'unità ad ogni ciclo.
P(1), P(2) - Punteggi dei due contendenti.
F(1), F(2) - Atteggiamento o posizione assunta dai pugili. Il valore 1 indica una guardia alta mentre il valore 2 una guardia bassa.
ROUND - Memorizza il numero del round.
PG - Assume i due valori, 1 o 2, in riferimento al pugile cui spetta il controllo in un dato istante.
W - Variabile di controllo. Ad ogni ciclo cambia il suo valore da -1 a +1 controllando il contenuto della variabile PG.

FACILI CAMBIAMENTI

COLORE DEGLI ELEMENTI SCENOGRAFICI. Con un apparecchio televisivo in bianco e nero si impone l'uso di colori contrastanti dal momento che molti risultano quasi indistinguibili tra di loro. Con un televisore a colori invece si possono utilizzare tutti gli abbinamenti con i 16 colori. In questo caso si devono modificare le linee 310 e 490.

COLORE DEI PUGILI. Essendo questi degli sprite, a nulla servono le istruzioni CALL COLOR. Il colore, in questo caso, è definito dalla terza costante contenuta tra parentesi nell'istruzione CALL SPRITE. Si ricorda che il pugile di sinistra è associato agli sprite #10 e #11, mentre i suoi calzoncini sono lo sprite #9; il pugile di destra è dato invece dagli sprite #20 e #21, mentre i suoi calzoncini sono lo sprite #19. Pertanto le linee da modificare sono tutte quelle che ad essi si riferiscono.

DURATA DEI ROUND. La durata dei round è determinata dall'istruzione IF THEN contenuta alla linea 900. Attualmente ogni ripresa termina quando TT raggiunge il valore 200. Scrivendo al suo posto un numero maggiore, ovviamente, si allunga la durata e viceversa.

DETERMINAZIONE DEL PUNTEGGIO. Attualmente il punteggio è stabilito alla linea 880. Semplicemente si assegna un punto se viene tirato un colpo alla figura mentre l'avversario è in guardia alta, oppure se un colpo al volto trova l'avversario in guardia bassa. Si può sostituire la linea 880 con una routine che determini il punteggio in base a criteri più complessi.

PRIMO ATTERRAMENTO. Il primo atterramento si verifica quando uno dei due pugili abbia subito 100 colpi validi ai fini del punteggio. Generalmente ciò non si verifica prima di 8-10 riprese. Se si desidera accelerare i tempi basta modificare la linea 890 scrivendo, ad esempio: IF P(PG) >50 THEN 990.

NUMERO DEGLI ATTERRAMENTI. Quando si verifica un atterramento i punteggi dei due pugili vengono diminuiti entrambi di 10 unità e l'incontro prosegue. Dopo 5 atterramenti, di uno o di entrambi i pugili, l'incontro termina per KO. Cambiando la linea 1130 si può consentire un numero maggiore o minore di atterramenti.

X, Y - Variabili di stato dei joystick. In base al loro valore si produce una particolare azione.
Z1, Z2 - Variabili di lavoro contenenti i codici ASCII dei caratteri definiti via software, rappresentanti i due pugili.
TIME - Variabile di controllo dei loop per creare delle pause.
KD - Numero degli atterramenti subiti dai due pugili.
FLAG - Variabile flag che assume il valore 1 al termine delle 12 riprese. A differenza del KO, in questo caso, è previsto il pareggio.
K, S - Usate nell'istruzione CALL KEY, indicano rispettivamente il numero di codice ASCII del tasto premuto e lo stato della tastiera.
J - Variabile di lavoro per il controllo dei cicli FOR NEXT.

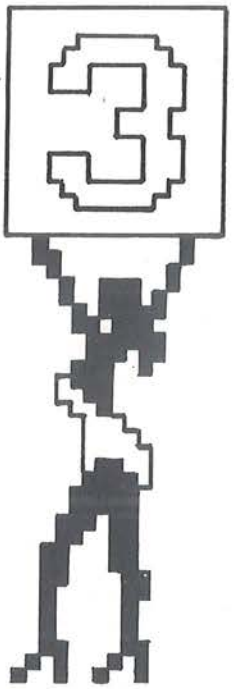


Figura 3. *La tecnica usata con i caratteri grafici definiti via software consente una certa libertà e limiti meno angusti. Nel programma non si è voluto rinunciare nemmeno alla ragazza che, durante l'intervallo, porta il cartello con il numero del round seguente.*

in un colpo d'occhio...

...il panorama completo di tutte le attività italiane.

l'Annuario Generale Italiano
viene letto da milioni di esponenti politici, governativi, amministrativi, industriali, bancari e finanziari.
Viene consultato da coloro che studiano, riflettono, valutano ed alla fine... decidono.

GUIDA MONACI
ANNUARIO GENERALE ITALIANO

00187 ROMA - Via F. Crispi, 10 - Tel. 06/483.401 - Telex 613462 MONACI
20145 MILANO - Via V. Monti, 86 - Tel. 02/345.8567 - Telex 332849 MONACI

PERSONAL SOFTWARE

Roulette

Un classico gioco per il vostro Texas TI99

di *Mauro Cristuib Grizzi*

Questo programma è stato scritto per il TI99/4A fornito di modulo SSS Extended BASIC, e permette di giocare alla roulette europea (quella diffusa nei nostri Casinò e senza il doppio zero).
Possono esserci fino a sei giocatori contemporaneamente, mentre le giocate possibili sono le seguenti:

- Numeri Pieni: è la classica giocata su un numero che, se estratto, rende 35 volte la posta;
- Cavalli: gruppi di due numeri, ad esempio 1 e 2 oppure 1 e 4; paga 17 volte la posta;
- Terzine: gruppi di tre numeri consecutivi sulla stessa riga, come 1,2,3 oppure 13,14,15; paga 11 volte la posta;
- Carré: gruppi di quattro numeri in quadrato, come 1, 2, 3, 5 oppure 14, 15, 17, 18; paga 8 volte la posta;
- Sestine: gruppi di sei numeri consecutivi su due righe, come sestina 1-6, oppure sestina 13-18; paga 5 volte la posta;
- Dozzine: le tre dozzine giocabili sono la 1-12, la 13-24 e la 25-36; pagano 2 volte la posta;
- Colonne: le tre colonne A, B, e C pagano 2 volte la posta;
- Pari/Dispari e Rosso/Nero pagano 1 volta la posta.

Il programma, dopo essersi informato sul numero dei giocatori e sui rispettivi capitali disponibili, passa al primo giocatore e gli chiede quali giocate intenda fare tramite un menu di gioco. Scegliendo la (o le) opzioni desiderate, si visualizzerà il tavolo verde ed il programma chiederà numeri e puntata. Per puntate su

Listato 1. *Il listato del programma Roulette, è stato ottenuto usando una speciale codifica per i caratteri di controllo: quando trovate in una stringa un carattere sottolineato, ciò significa che dovrete premere quel tasto insieme al tasto Control. Ad esempio, A significa Control + A.*

```
100 GOTO 130 :: OPTION BASE
1 :: DIM M(34,6),MA(34,6),M2
(6,6),PU(7),FP(6),NOME$(6),C
L$(6),COL$(3,6):: CALL CHARS
ET :: CALL GCHAR
110 CALL CLEAR :: CALL SCREE
N :: CALL COLOR :: CALL SOUN
D :: CALL KEY :: CALL CHAR :
: CALL CHARPAT :: CALL VCHAR
 :: CALL HCHAR :: RANDOMIZE

120 PLAYER,PL,GI,I,W,N,A,X,K
EY,STATUS,A$,B$,C$ :: !@P-
130 GOSUB 1390 :: CALL CLEAR
 :: CALL CHARSET
140 FOR I=49 TO 56 :: CALL C
HARPAT(I,A$):: CALL CHAR(I+6
3,A$,I+71,A$):: NEXT I :: CA
LL CHARPAT(48,A$,57,B$):: CA
LL CHAR(102,A$,103,B$,110,A$
,111,B$)
150 CALL COLOR(0,16,1,1,8,1,
7,7,15,9,7,15,10,2,15,11,7,1
5,12,2,15,14,8,15)
160 CALL CHAR(130,"000000000
000000080808080808080800000
0FF"):: CALL CHAR(136,"00000
0FF")
170 RANDOMIZE :: FOR I=5 TO
8 :: CALL COLOR(I,8,1):: NEX
T I :: FOR I=2 TO 4 :: CALL
COLOR(I,16,1):: NEXT I
180 PRINT "HHHHHHHHHHHHHHHHH
HHHHHHHHHH": :" INSERIRE LE
 GIOCATE,TRANNE  LA PUGLIA,I
N UNITA'DI 1000": :"HHHHHHHH
HHHHHHHHHHHHHHHHHHHHHH": : : :
 : : : : : : : : : :
 : : : :
190 INPUT "QUANTI GIOCATORI?
(MAX.6)":PLAYER :: FOR PL=1
TO PLAYER :: INPUT "NOME,PRE
GO? ":NOME$(PL):: INPUT "PUG
LIA DI PARTENZA?":PU(PL):: P
RINT :: NEXT PL :: G
I=1
200 FOR PL=1 TO PLAYER :: IF
 PL=FP(PL)THEN 500
210 PU(7)=0 :: CL$(PL)="" ::
 FOR I=1 TO 3 :: COL$(I,PL)=
"" :: NEXT I :: FOR I=1 TO 3
4
220 MA(I,PL),M(I,PL)=0 :: NE
XT I :: FOR I=1 TO 6 :: M2(I
,PL)=0 :: NEXT I :: GOSUB 13
30
230 CALL KEY(0,KEY,STATUS)::
 IF STATUS=0 OR(KEY<48 OR KE
Y>57)THEN 230
240 KEY=KEY-48 :: IF KEY=0 T
HEN 470
250 ON KEY GOSUB 270,290,310
,330,350,380,410,430,450
260 GOSUB 1330 :: GOTO 230
270 GOSUB 1110 :: INPUT "QUA
NTI PIENI?(MAX.6) ":N :: FOR
 I=1 TO N :: INPUT "NUMERO?P
UNTATA?":M(I,PL),A :: MA(I,P
L)=A*1000
280 PU(7)=PU(7)+MA(I,PL):: N
EXT I :: RETURN
290 GOSUB 1110 :: INPUT "QUA
NTI CAVALLI?(MAX.6) ":N :: F
OR I=1 TO N :: INPUT "NUMERI
?PUNTATA?":M(I+6,PL),M2(I,PL
),A
300 MA(I+6,PL)=A*1000 :: PU(
7)=PU(7)+MA(I+6,PL):: NEXT I
 :: RETURN
310 GOSUB 1110 :: INPUT "QUA
NTI CARRE'?(MAX.6) ":N :: FO
R I=1 TO N :: INPUT "NUM.PIU
'BASSO?PUNTATA?":M(I+12,PL),
A
320 MA(I+12,PL)=A*1000 :: PU
(7)=PU(7)+MA(I+12,PL):: NEXT
 I :: RETURN
330 GOSUB 1110 :: INPUT "QUA
NTE TERZINE?(MAX.6) ":N :: F
OR I=1 TO N :: INPUT "NUM.PI
U'BASSO?PUNTATA?":M(I+18,PL)
,A :: MA(I+18,PL)=A*1000
340 PU(7)=PU(7)+MA(I+18,PL):
: NEXT I :: RETURN
```

Terzine, Carré, Sestine e Dozzine sarà sufficiente battere il numero più basso compreso nel gruppo, ed il computer provvederà a calcolare anche gli altri automaticamente. Dopo ogni puntata viene visualizzato il capitale ancora disponibile e quello giocato; finito di puntare, occorre premere il tasto di fine giocata ed il calcolatore, dopo avervi informato sul totale giocato, passerà a raccogliere le puntate del giocatore seguente.

Quando tutti avranno puntato, verrà estratto il fatidico numero e visualizzati gli esiti, giocatore per giocatore e giocata per giocata. Si passerà quindi a raccogliere le puntate successive.

Il programma è ricco di controlli e quindi, se provate a barare, vi prenderà sempre in "castagna". Se poi malauguratamente doveste finire i soldi, il computer ve ne presterà quanti ne vorrete, e senza pretenderne la restituzione!

```
350 GOSUB 1110 :: INPUT "QUA
NTE SESTINE?(MAX.6)":N :: FO
R I=1 TO N :: INPUT "NUM.PIU
'BASSO?PUNTATA?":M(I+24,PL),
A :: FOR W=1 TO 34 STEP 3 ::
 IF M(I+24,PL)=W THE
N 370 ELSE 360
360 NEXT W :: PRINT "NUMERO
NON CONGRUENTE" :: FOR W=1 T
O 150 :: NEXT W :: GOTO 350

370 MA(I+24,PL)=A*1000 :: PU
(7)=PU(7)+MA(I+24,PL):: NEXT
 I :: RETURN
380 GOSUB 1110 :: INPUT "QUA
NTE DOZZINE?":N :: FOR I=1 T
O N :: INPUT "NUM.PIU'BASSO?
PUNTATA?":M(I+28,PL),A :: FO
R W=1 TO 25 STEP 12 :: IF M(
I+28,PL)=W THEN 400
ELSE 390
390 NEXT W :: PRINT "NUMERO
NON CONGRUENTE" :: FOR W=1 T
O 150 :: NEXT W :: GOTO 380

400 MA(I+28,PL)=A*1000 :: PU
(7)=PU(7)+MA(I+28,PL):: NEXT
 I :: RETURN
410 CALL CLEAR :: INPUT "QUA
NTE COLONNE?":N :: PRINT ::
FOR I=1 TO N :: INPUT "COLON
NA?PUNTATA?":COL$(I,PL),A ::
 MA(I+30,PL)=A*1000
420 PU(7)=PU(7)+MA(I+30,PL):
: NEXT I :: RETURN
430 CALL CLEAR :: INPUT "PAR
I-DISPARI?(P/D)PUNTATA?  ":A
$,A :: IF A$="P".THEN COL$(3
,PL)="PARI" ELSE COL$(3,PL)=
"DISPARI"
440 MA(33,PL)=A*1000 :: PU(7
)=PU(7)+MA(33,PL):: RETURN
450 CALL CLEAR :: INPUT "COL
ORE (R/N)?PUNTATA?":A$,A ::
IF A$="R" THEN CL$(PL)="ROSS
O" ELSE CL$(PL)="NERO"
460 MA(34,PL)=A*1000 :: PU(7
)=PU(7)+MA(34,PL):: RETURN
470 DISPLAY AT(24,1):USING "
TOT GIOCATO      #######":PU(
7)
480 FOR W=1 TO 500 :: NEXT W
 :: IF PU(7)<=PU(PL)THEN 500

490 CALL SOUND(100,500,0)::
PRINT "GIOCATA SUPERIORE A Q
UANTO  POSSEDUTO----NON ACCE
TTATA" :: FOR I=1 TO 500 ::
NEXT I :: GOTO 210
500 NEXT PL :: PRINT : : ::
X=INT(37*RND):: CALL CLEAR :
: FOR I=1 TO 10 :: DISPLAY A
T(16,5)BEEP:"ATTENDERE PREGO
..."
510 FOR W=1 TO 50 :: NEXT W
:: CALL CLEAR :: FOR W=1 TO
50 :: NEXT W :: NEXT I
520 FOR I=1 TO 35 STEP 2 ::
IF X=I THEN B$="NERO" :: GOT
O 540
530 NEXT I :: IF X=0 THEN B$
="---" ELSE B$="ROSSO"
540 FOR I=2 TO 36 STEP 2 ::
IF X=I THEN C$="PARI" :: GOT
O 560
550 NEXT I :: IF X=0 THEN C$
="---" ELSE C$="DISP."
560 FOR PL=1 TO PLAYER :: PU
(7)=0 :: IF PL=FP(PL)THEN 10
80
570 PRINT "USCITO IL";X;"-"&
C$&"-"&B$&"-": : :: CALL SOU
ND(200,1500,0):: PRINT "RISU
LTATI DI ";NOME$(PL): : :: F
OR I=1 TO 6 :: IF MA(I,PL)=0
 THEN 610
580 IF X<>M(I,PL)THEN 600
590 PRINT "PIENO   ";M(I,PL)
;TAB(23);"*VINCE" :: PU(7)=P
U(7)+35*MA(I,PL):: GOTO 610

600 PRINT "PIENO   ";M(I,PL)
;TAB(23);"PERDE" :: PU(7)=PU
(7)-MA(I,PL)
610 NEXT I :: FOR I=7 TO 12
:: IF MA(I,PL)=0 THEN 650
620 IF X=M(I,PL)OR X=M2(I-6,
PL)THEN 640

630 PRINT "CAVALLO ";M(I,PL)
;M2(I-6,PL);TAB(23);"PERDE"
:: PU(7)=PU(7)-MA(I,PL):: GO
TO 650
640 PRINT "CAVALLO ";M(I,PL)
;M2(I-6,PL);TAB(23);"*VINCE"
 :: PU(7)=PU(7)+17*MA(I,PL)

650 NEXT I :: FOR I=13 TO 18
 :: IF MA(I,PL)=0 THEN 690
660 A=M(I,PL):: IF X=A OR X=
A+1 OR X=A+3 OR X=A+4 THEN 6
80
```

Alcuni commenti al listato

100-120 - Elenco dei sottoprogrammi e delle variabili usate, per ridurre il tempo di attesa dopo il Run.

130-160 - Presentazione e definizione caratteri speciali.

170-190 - Raccolta dei dati su numero di giocatori, rispettivi nomi e capitali disponibili.

200-260 - Visualizzazione del menu di gioco ed accettazione della opzione selezionata.

270-280 - Subroutine accettazione Numeri Pieni.

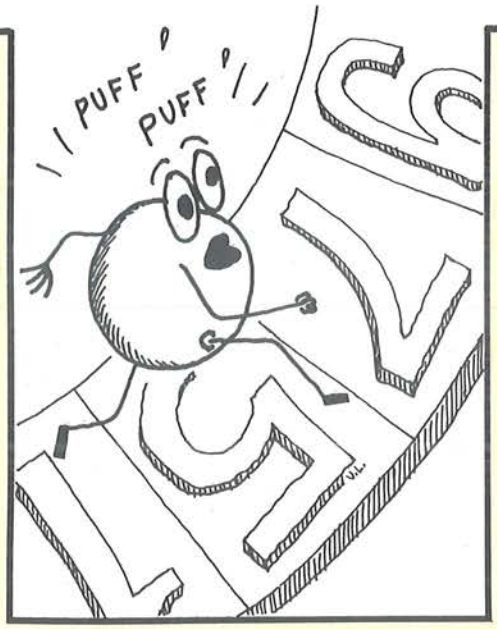
290-300 - Subroutine accettazione Cavalli.

310-320 - Subroutine accettazione Carré.

330-340 - Subroutine accettazione Terzine.

Seguito listato Roulette.

```
670 PRINT "CARRE' ";A;A+1;A+
3;A+4;TAB(23);"PERDE" :: PU(
7)=PU(7)-MA(I,PL):: GOTO 690

680 PRINT "CARRE' ";A;A+1;A+
3;A+4;TAB(23);"*VINCE" :: PU
(7)=PU(7)+8*MA(I,PL)
690 NEXT I :: FOR I=31 TO 32
 :: IF MA(I,PL)=0 THEN 760
700 FOR W=1 TO 34 STEP 3 ::
IF W=X THEN A$="A"
710 IF X=W+1 THEN A$="B"
720 IF X=W+2 THEN A$="C"
730 NEXT W :: IF A$=COL$(I-3
0,PL)THEN 750
740 PRINT "COLONNA       ";CO
L$(I-30,PL);TAB(23);"PERDE"
:: PU(7)=PU(7)-MA(I,PL):: GO
TO 760
750 PRINT "COLONNA       ";CO
L$(I-30,PL);TAB(23);"*VINCE"
 :: PU(7)=PU(7)+2*MA(I,PL)
760 NEXT I :: FOR I=19 TO 24
 :: IF MA(I,PL)=0 THEN 800
770 A=M(I,PL):: IF X=A OR X=
A+1 OR X=A+2 THEN 790
780 PRINT "TERZINA ";A;A+1;A
+2;TAB(23);"PERDE" :: PU(7)=
PU(7)-MA(I,PL):: GOTO 800
790 PRINT "TERZINA ";A;A+1;A
+2;TAB(23);"*VINCE" :: PU(7)
=PU(7)+11*MA(I,PL)
800 NEXT I :: FOR I=25 TO 28
 :: IF MA(I,PL)=0 THEN 840
810 FOR W=M(I,PL)TO M(I,PL)+
5 :: IF X<>W THEN 830
820 PRINT "SESTINA ";W;"-";W
+5;TAB(23);"*VINCE" :: PU(7)
=PU(7)+5*MA(I,PL):: GOTO 840

830 NEXT W :: PRINT "SESTINA
 ";M(I,PL);"-";M(I,PL)+5;TAB
(23);"PERDE" :: PU(7)=PU(7)-
MA(I,PL)
840 NEXT I :: FOR I=29 TO 30
 :: IF MA(I,PL)=0 THEN 880
850 IF X<M(I,PL)OR X>M(I,PL)
+11 THEN 870
860 PRINT "DOZZINA ";M(I,PL)
;"-";M(I,PL)+11;TAB(23);"*VI
NCE" :: PU(7)=PU(7)+2*MA(I,P
L):: GOTO 880
870 PRINT "DOZZINA ";M(I,PL)
;"-";M(I,PL)+11;TAB(23);"PER
DE" :: PU(7)=PU(7)-MA(I,PL)

880 NEXT I :: IF MA(33,PL)=0
 THEN 970
890 IF X=0 THEN 960
900 IF COL$(3,PL)="PARI" THE
N A=1 ELSE A=2
```




```
910 FOR I=2 TO 36 STEP 2 ::
IF X=I THEN 920 ELSE 930
920 ON A GOTO 950,960
930 NEXT I
940 ON A GOTO 960,950
950 PRINT COL$(3,PL);TAB(23)
;"*VINCE" :: PU(7)=PU(7)+MA(
33,PL):: GOTO 970
960 PRINT COL$(3,PL);TAB(23)
;"PERDE" :: PU(7)=PU(7)-MA(3
3,PL)
970 IF MA(34,PL)=0 THEN 1010

980 FOR I=1 TO 35 STEP 2 ::
IF X=I AND CL$(PL)="NERO" TH
EN PRINT CL$(PL);TAB(23);"*V
INCE" :: PU(7)=PU(7)+MA(34,P
L):: GOTO 1010
990 NEXT I :: FOR I=2 TO 36
STEP 2 :: IF X=I AND CL$(PL)
="ROSSO" THEN PRINT CL$(PL);
TAB(23);"*VINCE" :: PU(7)=PU
(7)+MA(34,PL):: GOTO 1010
1000 NEXT I :: PRINT CL$(PL)
;TAB(23);"PERDE" :: PU(7)=PU
(7)-MA(34,PL)
1010 PU(PL)=PU(PL)+PU(7)
1020 PRINT USING "ESITO GIOC
ATA +##########":PU(7):: PRI
NT USING "NUOVA PUGLIA  ####
########":PU(PL):: IF PU(PL)>
0 THEN 1060
1030 CALL SOUND(1000,140,0):
: PRINT : :"HAI FINITO I SOL
DI!": : :: INPUT "VUOI UNA N
UOVA PUGLIA? ":A$ :: IF A$<>
"SI" THEN 1050
1040 INPUT "QUANTO? ":PU(PL)
:: GOTO 1060
1050 FP(PL)=PL
```

```
1060 PRINT : :"****** PREMI
UN TASTO ******": : :
1070 CALL KEY(0,KEY,STATUS):
: IF STATUS=0 THEN 1070 ELSE
 1080
1080 NEXT PL
1090 GI=GI+1 :: GOTO 200
1100 END
1110 CALL CLEAR :: CALL COLO
R(13,2,15)
1120 FOR I=5 TO 14 :: CALL V
CHAR(8,I,130,16):: NEXT I ::
 FOR I=19 TO 28 :: CALL VCHA
R(10,I,130,14):: NEXT I
1130 FOR I=8 TO 11 STEP 3 ::
 CALL VCHAR(9,I,131,15):: NE
XT I
1140 FOR I=22 TO 25 STEP 3 :
: CALL VCHAR(10,I,131,14)::
NEXT I
1150 FOR I=10 TO 22 STEP 2 :
: CALL HCHAR(I,6,132,8):: CA
LL HCHAR(I,20,132,8):: NEXT
I :: CALL HCHAR(18,6,136,8):
: CALL HCHAR(14,20,136,8)
1160 DISPLAY AT(9,4)SIZE(-8)
:"BBBBnBBB" :: DISPLAY AT(11
,4)SIZE(-2):"Bx"
1170 DISPLAY AT(11,7)SIZE(-2
):"Bq" :: DISPLAY AT(11,10)S
IZE(-2):"Bz" :: DISPLAY AT(1
3,4)SIZE(-2):"Bs"
1180 DISPLAY AT(13,7)SIZE(-2
):"B|" :: DISPLAY AT(13,10)S
IZE(-2):"Bu" :: DISPLAY AT(1
5,4)SIZE(-2):"B~"
1190 DISPLAY AT(15,7)SIZE(-2
):"Bw" :: DISPLAY AT(15,10)S
IZE(-2):"Bo" :: DISPLAY AT(1
7,4)SIZE(-2):"pf"
1200 DISPLAY AT(17,7)SIZE(-2
):"xx" :: DISPLAY AT(17,10)S
IZE(-2):"pq"
1210 DISPLAY AT(19,4)SIZE(-2
):"xz" :: DISPLAY AT(19,7)SI
ZE(-2):"ps" :: DISPLAY AT(19
,10)SIZE(-2):"x|"
1220 DISPLAY AT(21,4)SIZE(-2
):"pu" :: DISPLAY AT(21,7)SI
ZE(-2):"x~" :: DISPLAY AT(21
,10)SIZE(-2):"pw"
1230 DISPLAY AT(23,4)SIZE(-2
):"BA" :: DISPLAY AT(23,7)SI
ZE(-2):"BB" :: DISPLAY AT(23
,10)SIZE(-2):"BC"
1240 DISPLAY AT(11,18)SIZE(-
2):"xo" :: DISPLAY AT(11,21)
SIZE(-2):"qf"
1250 DISPLAY AT(11,24)SIZE(-
2):"yx" :: DISPLAY AT(13,18)
SIZE(-2):"qq" :: DISPLAY AT(
```

```
13,21)SIZE(-2):"yz"
1260 DISPLAY AT(13,24)SIZE(-
2):"qs" :: DISPLAY AT(15,18)
SIZE(-2):"y|"
1270 DISPLAY AT(15,21)SIZE(-
2):"qu" :: DISPLAY AT(15,24)
SIZE(-2):"y~" :: DISPLAY AT(
17,18)SIZE(-2):"qw"
1280 DISPLAY AT(17,21)SIZE(-
2):"yo" :: DISPLAY AT(17,24)
SIZE(-2):"rf"
1290 DISPLAY AT(19,18)SIZE(-
2):"zx" :: DISPLAY AT(19,21)
SIZE(-2):"rq" :: DISPLAY AT(
19,24)SIZE(-2):"zz"
1300 DISPLAY AT(21,18)SIZE(-
2):"rs" :: DISPLAY AT(21,21)
SIZE(-2):"z|" :: DISPLAY AT(
21,24)SIZE(-2):"ru"
1310 DISPLAY AT(23,18)SIZE(-
2):"BA" :: DISPLAY AT(23,21)
SIZE(-2):"BB" :: DISPLAY AT(
23,24)SIZE(-2):"BC"
1320 PRINT :: RETURN
1330 CALL CLEAR :: CALL COLO
R(13,9,2):: DISPLAY AT(1,1):
"GIOCATA N.";GI;"DI";" "&NOM
E$(PL):: DISPLAY AT(2,1):"DD
DDDDDDDDDDDDDDDDDDDDDDDDDD"

1340 DISPLAY AT(3,1):"PRESS
1 PER PIENI" :: DISPLAY AT(5
,1):"PRESS 2 PER CAVALLI"
1350 DISPLAY AT(7,1):"PRESS
3 PER CARRE'" :: DISPLAY AT(
9,1):"PRESS 4 PER TERZINE" :
: DISPLAY AT(11,1):"PRESS 5
PER SESTINE" :: DISPLAY AT(1
3,1):"PRESS 6 PER DO
ZZINE"
1360 DISPLAY AT(15,1):"PRESS
 7 PER COLONNE" :: DISPLAY A
T(17,1):"PRESS 8 PER PARI/DI
SP." :: DISPLAY AT(19,1):"PR
ESS 9 PER ROSSO/NERO"
1370 DISPLAY AT(21,1):"PRESS
 0 PER FINE" :: DISPLAY AT(2
3,1):"POSSEDUTE" :: DISPLAY
AT(23,17):USING "#######":PU
(PL)
1380 DISPLAY AT(24,1):"DISP.
DA GIOCARE" :: DISPLAY AT(24
,17):USING "#######":PU(PL)-
PU(7):: RETURN
1390 FOR I=65 TO 88 :: CALL
CHARPAT(I,A$):: CALL CHAR(I+
31,A$):: NEXT I :: FOR I=9 T
O 11 :: CALL COLOR(I,2,11)::
 NEXT I
1400 CALL CLEAR :: CALL SCRE
EN(2):: CALL CHAR(136,""):::
CALL COLOR(14,9,9):: FOR I=2
 TO 32 :: CALL HCHAR(1,I,136
):: NEXT I :: FOR I=1 TO 24
:: CALL VCHAR(I,32,1
36)
1410 NEXT I :: FOR I=32 TO 2
 STEP -1 :: CALL HCHAR(24,I,
136):: NEXT I :: FOR I=24 TO
 1 STEP -1 :: CALL VCHAR(I,2
,136):: NEXT I :: CALL CHAR(
129,""):: CALL COLOR
(13,5,5):: W=3 :: PL=13 :: X
=5 :: N=29
1420 FOR I=X TO N :: CALL HC
HAR(W,I,129):: NEXT I :: FOR
 A=W TO PL :: CALL VCHAR(A,N
,129):: NEXT A :: FOR I=N TO
 X STEP -1 :: CALL HCHAR(PL,
I,129)
1430 NEXT I :: FOR A=PL TO W
 STEP -1 :: CALL VCHAR(A,X,1
29):: NEXT A :: X=X+1 :: N=N
-1 :: W=W+1 :: PL=PL-1 :: IF
 PL-W<0 THEN 1440 ELSE 1420

1440 FOR W=3 TO 8 :: CALL CO
LOR(W,16,5):: NEXT W :: A$="
0613ROULETTE" :: GOSUB 1540

1450 A$="0810BYAMACRISTUIB"
:: GOSUB 1540 :: CALL CHAR(1
20,""):: CALL COLOR(12,11,11
):: CALL CHAR(58,"FFFFFFFFFF
FFFFFF")
1460 FOR I=17 TO 19 STEP 2 :
: FOR W=8 TO 26 :: CALL HCHA
R(I,W,120):: NEXT W :: FOR W
=26 TO 8 STEP -1 :: CALL HCH
AR(I+1,W,120):: NEXT W :: NE
XT I
1470 A$="1809sdw`rxhmrsqtldm
sr" :: GOSUB 1540 :: A$="191
3bnlotsdq" :: GOSUB 1540 ::
FOR A=24 TO 1 STEP -2 :: FOR
 X=2 TO 32 :: CALL GCHAR(A,X
,W):: IF W=32 THEN C
ALL HCHAR(A,X,58)
1480 IF W=129 THEN X=29
1490 IF W=120 THEN X=26
1500 NEXT X :: FOR X=32 TO 2
 STEP -1 :: CALL GCHAR(A-1,X
,W):: IF W=32 THEN CALL HCHA
R(A-1,X,58)
1510 IF W=129 THEN X=5
1520 IF W=120 THEN X=8
1530 NEXT X :: NEXT A :: RET
URN
1540 A=VAL(SEG$(A$,1,2)):: X
=VAL(SEG$(A$,3,2)):: FOR I=5
 TO LEN(A$):: W=ASC(SEG$(A$,
I,1)):: CALL HCHAR(A,X,W)::
CALL SOUND(5,800,0):: X=X+1
:: NEXT I :: RETURN
```

350-370 - Subroutine accettazione Sestine e controllo correttezza dei numeri scelti.

380-400 - Subroutine accettazione Dozzine e controllo correttezza dei numeri scelti.

410-420 - Subroutine accettazione Colonne.

430-440 - Subroutine accettazione Pari/Dispari.

450-460 - Subroutine accettazione Rosso/Nero.

470-490 - Visualizzazione del totale giocato e controllo che questo non superi il capitale disponibile.

500-510 - Ciclo di attesa prima dell'estrazione del numero vincente.

520-550 - Calcolo del colore del numero estratto e verifica della sua divisibilità per due (Pari o Dispari).

560 - Ciclo For/Next per il numero totale di giocatori, con controllo sui giocatori che hanno esaurito il proprio capitale e vanno quindi "saltati" dal ciclo stesso.

570 - Presentazione del numero uscito ed intestazione dei risultati della giocata del giocatore PL.

580-600 - Analisi della giocata sui Pieni.

610-650 - Analisi della giocata sui Cavalli.

660-690 - Analisi della giocata sui Carré.

700-760 - Analisi della giocata sulle Colonne.

770-800 - Analisi della giocata sulle Terzine.

810-840 - Analisi della giocata sulle Sestine.

850-880 - Analisi della giocata sulle Dozzine.

890-1010 - Analisi della giocata su Pari/Dispari e Rosso/Nero.

1020-1080 - Calcolo e visualizzazione di esito giocata, nuovo capitale e, nel caso questo sia nullo, offerta di una nuova puglia.

1110-1320 - Subroutine visualizzazione del tavolo di gioco.

1330-1380 - Subroutine visualizzazione menu di gioco.

1390-1540 - Subroutine presentazione del gioco. ■

TEXAS TI99/4A

Recupero degli errori

di Sergio Borsani

Un buon programma dovrebbe essere in grado di sopravvivere a tutti i tentativi, più o meno consapevoli, di portarlo in una condizione d'errore. In un certo senso il programmatore deve prevedere ogni possibile risposta, da parte dell'utente, che possa causare la comparsa di un messaggio d'errore e il conseguente blocco del programma. In questo compito, non sempre facile, il TI Extended BASIC viene in aiuto con due istruzioni: On Error e Call Err.
La prima esiste in due formati: On Error Stop e On Error n, dove n è un appropriato numero di linea. La parola chiave Stop ha valore di default e fa in modo che appaia il solito messaggio dopo il quale il programma si arresta; nè più nè meno di quello che avviene normalmente. Ben diversamente agisce il secondo formato. Esso permette di inviare il controllo ad una subroutine, come fa una Gosub, in modo da poter visualizzare un qualsiasi messaggio e poi procedere nell'esecuzione del programma.
L'istruzione On Error n deve essere ripetuta ogni volta che si incappa in un errore per cui il punto più ovvio dove inserirla in un programma è il menu principale che costituisce un passaggio obbligato. La routine che gestisce l'errore, come tutte le subroutine, termina con l'istruzione Return. Tuttavia, a differenza di quanto avviene con una Gosub, lo statement Return può assumere tre formati: Return da solo, Return + numero di linea e Return Next. Return fa tornare il programma alla linea che ha causato l'errore. Return + numero di linea trasferisce il controllo a quel numero di linea. Return Next fa riprendere la esecuzione del programma dalla linea successiva a quella che ha causato l'errore. Nel caso si specifichi un numero di linea, il programma potrebbe opportunamente tornare al menu dove verrebbe eseguita nuovamente l'istruzione On Error prima di proseguire in una qualsiasi sezione di lavoro.
Naturalmente questa è solo una proposta e nulla vieta una diversa strutturazione.

La subroutine che viene chiamata in caso di errore, invece di limitarsi ad inviare un messaggio generico, può essere potenziata e resa in grado di specificare all'utente il tipo di errore e la linea di programma nella quale si è verificato. Ciò è reso possibile dall'uso dell'istruzione Call Err. La sintassi completa è: Call Err (codice, tipo [,x,numero di linea]). Quando nel programma viene commesso un errore e viene eseguita una Call Err la prima variabile numerica tra parentesi contiene il codice dell'errore. Il suo significato è reso esplicito nell'Appendice N del manuale del BASIC esteso, a partire da pag. 212. Se si caricano in una tabella alcuni codici ed i corrispondenti messaggi d'errore, la subroutine diventa in grado di indicare all'utente il tipo di errore che è stato commesso. La seconda variabile tra parentesi assume solitamente un valore negativo. Se il codice d'errore è il 130, allora essa indica il numero del file causa dell'errore. La terza variabile ha sempre il valore 9.
La quarta, infine, specifica il numero della linea di programma nella quale si è verificato l'errore. Le ultime due variabili tra parentesi sono facoltative, pertanto

Listato 1. *Il programma Recupero degli errori.*

```
100 REM ******************
110 REM *                  *
120 REM *    RECUPERO      *
130 REM *  DEGLI ERRORI    *
140 REM *                  *
150 REM ******************
160 DATA 109,FILE ERROR
170 DATA 24,STRING-NUMBER MISMATCH
180 DATA 14,SYNTAX ERROR
190 DATA 60,LINE NOT FOUND
200 DATA 130,I/O ERROR
210 FOR J=1 TO 5 :: READ EC(J),EM$(J)::
NEXT J
220 CALL CLEAR :: ON ERROR 540
230 DISPLAY AT(2,1):"IL PROGRAMMA E' DIV
ISO IN 6 SEZIONI OGNUNA CONTENENTE UNDIV
ERSO TIPO DI ERRORE."
240 DISPLAY AT(8,1):"SCEGLI LA SEZIONE."
250 FOR RIGA=10 TO 20 STEP 2 :: N=RIGA/2
-4 :: DISPLAY AT(RIGA,1):STR$(N);". SEZI
ONE ";CHR$(64+N):: NEXT RIGA
260 DISPLAY AT(24,10):"SCELTA?" :: ACCEP
T AT(24,18)SIZE(1)VALIDATE("123456"):SC$
270 ON VAL(SC$)GOTO 280,310,340,370,400,
430
280 CALL CLEAR :: DISPLAY AT(1,1):"QUEST
A SEZIONE CONTIENE UN  ERRORE DI INPUT/O
UTPUT." :: GOSUB 530
290 OPEN #4:"IEEE 488" :: PRINT #4:"TI-9
9/4A"
300 END
310 CALL CLEAR :: DISPLAY AT(1,1):"QUEST
A SEZIONE CONTIENE UNA ISTRUZIONE DOVE V
IENE CONFU-SA UNA STRINGA CON UN NUMERO"
 :: GOSUB 530
320 A$=99999
330 END
340 CALL CLEAR :: DISPLAY AT(1,1):"IN QU
ESTA SEZIONE SI TENTA  DI LEGGERE DATI D
A UN FILE  CHE NON ESISTE." :: GOSUB 530
350 INPUT #1:A$
360 END
370 CALL CLEAR :: DISPLAY AT(1,1):"IN QU
ESTA SEZIONE UNA GOSUB MANDA IL PROGRAMM
```

```
A AD UNA   LINEA INESISTENTE." :: GOSUB
530
380 GOTO 32767
390 END
400 CALL CLEAR :: DISPLAY AT(1,1):"IN QU
ESTA SEZIONE VIENE USA-TA COME VARIABILE
 UNA PAROLARISERVATA." :: GOSUB 530
410 MAX=999
420 END
430 CALL CLEAR :: DISPLAY AT(1,1):"IN QU
ESTA SEZIONE SI PUO'  VERIFICARE UNA CO
NDIZIONE DIERRORE CON LA FUNZIONE LOG."
440 DISPLAY AT(6,1):"ESSA SI VERIFICA QU
ANDO SI  PONE A<0."
450 DISPLAY AT(10,1):"SCRIVI UN NUMERO."
460 DISPLAY AT(12,1)BEEP:"A =" :: ACCEPT
 AT(12,5)VALIDATE(NUMERIC):A
470 L=LOG(A)
480 DISPLAY AT(14,1):"IL LOGARITMO DI";A
;"E'";L
490 DISPLAY AT(24,2):"- VUOI CONTINUARE?
 (Y/N) -"
500 CALL KEY(0,K,S):: IF S=0 THEN 500
510 IF K=89 OR K=121 THEN 430
520 IF K=78 OR K=110 THEN 220 ELSE 500
530 FOR TIME=1 TO 1000 :: NEXT TIME :: R
ETURN
540 CALL ERR(X,Y,W,Z):: FOR J=1 TO 5 ::
IF X=EC(J)THEN 560
550 NEXT J :: DISPLAY AT(20,1):"* ERRORE
!" :: GOTO 570
560 DISPLAY AT(20,1):"* ";EM$(J)
570 DISPLAY AT(21,3):"ALLA LINEA";Z
580 DISPLAY AT(24,6):"<PREMI UN TASTO>"
590 CALL KEY(0,K,S):: IF S=0 THEN 590 EL
SE RETURN 220
```

l'istruzione può essere scritta o nella forma Call Err (X, Y) oppure nell'altra Call Err (A,B,C,D).
Non è superfluo ricordare che le istruzioni On Error e Call Err formano un abbinamento veramente utile al programmatore nella fase di debug di un programma. Infatti, se si dovesse subire uno stop ad ogni errore, si dovrebbe ripartire con il Run azzerando così tutte le variabili presenti in quel momento con il risultato che si dovrebbero introdurre nuovamente i dati di prova. Recuperando gli errori si può invece proseguire con i controlli e le verifiche senza perdere tempo prezioso.
Il breve programma del listato 1 vuole semplicemente schematizzare i concetti su esposti mostrando come per alcuni errori possa essere mandato un messaggio specifico, per altri un messaggio generico. Il programma è strutturato in sei parti selezionabili dal menu; ognuna contiene un diverso errore.
Nell'ultima, la sesta, l'errore è conseguente al valore inserito dall'utente in input; infatti la funzione matematica Log, lì presente, non ammette argomenti negativi o uguali a zero.

Le istruzioni Data poste nella parte iniziale del programma contengono alcuni codici ed i corrispondenti messaggi d'errore. Alla linea 210 i dati vengono caricati in due vettori.
Le linee 220-270 costituiscono il menu e contengono l'istruzione On Error 540. Seguono le sei sezioni con altrettanti errori che altrimenti avrebbero causato uno stop del programma. La subroutine che li gestisce inizia alla linea 540 ed occupa tutta la parte restante, fino al termine. La stessa linea 540 contiene l'istruzione Call Err ed il ciclo per la ricerca in tabella del codice individuato dalla variabile X.

SINCLAIR ZX SPECTRUM

Extended BASIC

di Marcello Spero

Fra le possibilità offerte dall'Interfaccia 1, il cui scopo principale è quello di controller per i microdrive, c'è anche quella, meno nota, di estensione del vocabolario BASIC dello Spectrum.
Quello che vedremo questa volta è un metodo per ottenere il medesimo risultato senza l'ausilio dell'interfaccia.
È appena il caso di ricordare i vantaggi offerti dalla possibilità di disporre di un certo numero di istruzioni su misura, create apposta per risolvere i nostri problemi specifici.
Istruzioni per la grafica, tool kit, aiuti per la verifica e la correzione dei programmi, non sono che un pallido esempio.

Per ragioni che vedremo in seguito, tutti i nostri nuovi comandi ed istruzioni devono avere la caratteristica di provocare immediatamente (cioè con il loro primo carattere) un errore al normale controllo di sintassi operato dall'interprete in ROM. Due sono i metodi possibili per ottenere questo.
Uno, più semplice, consiste nello scegliere come primo carattere una delle parole BASIC, da Rnd a Step, ottenibili per mezzo del Symbol Shift. In questo modo si aggira il modo "K", e si ottiene un sicuro errore.
Il secondo metodo, più elegante ma più complesso, consiste nell'iniziare la nostra istruzione con un carattere che provochi l'uscita dal modo "K", e quindi comporre il nome della nuova istruzione lettera per lettera.

Un'istruzione che provoca lo "scrolling" laterale dello schermo, ad esempio, può essere realizzata con:

STEP d,5

(per uno spostamento verso destra di 5 pixel, ad esempio) con il primo metodo, o con:

PERSONAL SOFTWARE

Bioritmi

"Calcolare" il futuro con il TI99/4A

di *Daniele Sgorbini*

Tutto ciò che è misterioso suscita interesse e, pertanto non passa mai di moda, così la chiaroveggenza tiene il passo con i tempi, anche se ha dovuto adattarsi all'evoluzione degli stessi: la zingara col capo avvolto in un foulard che scruta la mitica sfera di cristallo ha lasciato il posto alla distinta signora che osserva i fosfori del video di un computer, nel nostro caso, di un Texas TI99/4A. Con questo programma si vuole permettere all'utente di predire scientificamente quello che potrà essere il suo stato di forma o di analizzare i motivi di una certa sua prestazione passata. Prima però di giungere alla descrizione dell'algoritmo del programma è necessario introdurre per sommi capi la teoria bioritmica. Questa teoria, pur non essendo universalmente accettata, presenta degli aspetti indubbiamente interessanti sui quali non esistono controversie: l'esistenza di cicli regolanti la nostra esistenza che ogni giorno abbiamo occasione di verificare tramite una semplice osservazione di ciò che ci circonda.

Si pensi ad esempio al ciclo giornaliero di apertura e chiusura dei petali di un fiore, all'alternarsi delle stagioni o al comportamento degli animali (letargo, accoppiamento, ecc.). Anche la vita umana è regolata da cicli; le ricerche fatte per cercare di identificarli sono, soprattutto nel campo della medicina, all'ordine del giorno: temperatura corporea, secrezioni glandolari e via di seguito. Il panorama è vasto ed affascinante e non si tratta di cose campate in aria, senza un fondamento: lo conferma il fatto che, nello studio dell'adattabilità dell'essere umano alla vita nello spazio o in condizioni particolarmente difficili, vengono utilizzati i bioritmi. Soprattutto i russi sono sensibili alle indicazioni che si

Listato 1. *Il programma Bioritmi.*

```
100 FOR I=1 TO 16
110 CALL COLOR(I,16,1)
120 NEXT I
130 CALL SCREEN(5)
140 CALL CHAR(150,"00383838FE7C3
810")
150 CALL CHAR(140,"000000FF01")

160 DEF RESTO(G)=((JV/G)-INT(JV/
G))*G
170 DIM X(23),Y(28),Z(32)
180 V$="didascaliedidascaliedida
scaliedidascaliedidascalliedidas
calie"
190 Y$="xxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxx"
200 K$="XXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXX"
210 NT$="** STATO INTELLETTIVO *
*"
220 PS$="**    STATO PSICHICO    *
*"
230 FS$="**    STATO FISICO      *
*"
240 MI$="MIN. FRA     GG"
250 MA$="MAX. FRA     GG"
260 FX$="FLEX.FRA     GG"
270 CALL CLEAR
280 PRINT "salve.":"questo progr
amma permette di conoscere lo st
ato bioritmico di una persona.":
::
290 INPUT "inserire la data di n
ascita del soggetto in analisi:
   ":G,M,A
300 GOSUB 1450
310 J1=JD
320 INPUT "inserire ora la data
del giorno da valutare:":G,M,A
330 GOSUB 1450
340 J2=JD
350 JV=J2-J1
360 CALL CLEAR
370 PR=33
380 MX=7
390 MN=22
400 FX=16
```

possono ricavare dallo studio dei bioritmi dei soggetti in esame: nella banca dati della NASA, infatti, alla voce bioritmi è elencata una serie di articoli, di cui molti di autori sovietici. Questi studi sono indirizzati particolarmente all'analisi delle variazioni dei cicli, per effetto dei campi magnetici o per effetto delle particolari condizioni che si verificano nei voli spaziali. Aspettarsi da un solo articolo una trattazione esauriente e profonda sulla teoria bioritmica è quindi una mera utopia; per approfondimenti si rimanda alla letteratura specializzata. Di conseguenza daremo per scontata la validità di ciò che verrà analizzato e ci limiteremo ad approfondirne i punti salienti.

Trascureremo le applicazioni più tecniche e più profonde, limitandoci alla descrizione di ciò che maggiormente può suscitare interesse tenendo conto anche del fatto che i lettori avranno conoscenze in campi molto disparati e che quindi una discussione molto tecnica e specifica sarebbe solamente tediosa per molti. Tratteremo solamente i cicli Fisico, Psichico ed Intellettivo, cioè quella branchia della teoria bioritmica che più è contestata, ma che più interessa il pubblico. Pur incontrando degli illustri denigratori, lo studio di questi cicli è proseguito rapidamente dando luogo a risultati e pareri contrastanti. In questo contesto di discordanze la mia opinione non potrà certo avere nessuna eco, ciononostante, vi posso assicurare che, facendo "girare" questo programma, si otterranno dei risultati sorprendenti. Almeno per quel che riguarda la mia cerchia familiare e le mie amicizie, le previsioni del computer si sono rivelate straordinariamente aderenti alla effettiva realtà delle cose. Se non siete convinti, provate ad implementare questo programma, anche solo per curiosità, e verificate la validità o meno dei risultati ottenuti.

I cicli della vita

I cicli che analizzeremo hanno un

```
410 GOSUB 1290
420 FOR I=1 TO LEN(NT$)
430 P=ASC(SEG$(NT$,I,1))
440 CALL HCHAR(1,3+I,P)
450 NEXT I
460 FOR I=1 TO LEN(MI$)
470 N=ASC(SEG$(MI$,I,1))
480 M=ASC(SEG$(MA$,I,1))
490 O=ASC(SEG$(FX$,I,1))
500 CALL HCHAR(3,15+I,N)
510 CALL HCHAR(5,15+I,M)
520 CALL HCHAR(7,15+I,O)
530 NEXT I
540 GOSUB 1650
550 GOSUB 1530
560 CALL HCHAR(13,1,140,32)
570 DATA 13,11,9,7,6,5,4,3,3,3,4
,5,6,7,9,11,13,15,17,19,20,21,22
,23,23,23,22,21,20,19,17,15
580 FOR I=1 TO 32
590 READ Z(I)
600 CALL HCHAR(Z(I),I,42)
610 NEXT I
615 IF R=32 THEN 630
620 CALL HCHAR(Z(R+1)-1,R+1,150)
630 CALL SOUND(90,700,2)
```

```
640 REM  CICLO EMO
650 CALL KEY(0,KEY,STATUS)
660 IF STATUS=0 THEN 650
670 REM  CANCELLA INT
680 FOR I=1 TO 32
690 CALL HCHAR(Z(I),I,32)
700 NEXT I
710 CALL HCHAR(14,1,32,312)
720 REM  RITRACCIA RIGA
730 CALL HCHAR(13,1,140,32)
740 CALL HCHAR(Z(R+1)-1,R+1,32)

750 FOR I=1 TO LEN(PS$)
760 P=ASC(SEG$(PS$,I,1))
770 CALL HCHAR(1,3+I,P)
780 NEXT I
790 CALL SCREEN(13)
800 MN=21
810 MX=7
820 FX=14
830 PR=28
840 GOSUB 1290
850 GOSUB 1650
860 V$=Y$
870 GOSUB 1530
880 DATA 12,9,7,5,4,3,2,2,3,4,5,
```

andamento assimilabile ad una curva del tipo sin(x) (figura 1). Questa curva (che prende il nome di sinusoide) descrive giorno per giorno il nostro stato di forma. Per chi fosse digiuno di geometria analitica spiegherò che la funzione sin(x) è una funzione che si ripete periodicamente nel tempo, cioè, all'aumentare di x essa si riconduce sempre ad un valore appartenente alla curva che descrive l'andamento della funzione. I cicli sono supposti essere in numero di tre: essi regolano il nostro stato fisico, psichico ed intellettivo ed hanno durate di 23, 28, 33 giorni rispettivamente. In ognuno di questi cicli sono individuabili 3 punti di particolare interesse: il minimo, il massimo ed il flesso. I primi due non necessitano di particolari commenti; il terzo che è il punto in cui la curva incontra l'asse delle ascisse riveste un'importanza tutta particolare in quanto, essendo il punto in cui il nostro bioritmo cambia di segno, è il giorno più critico pur non essendo

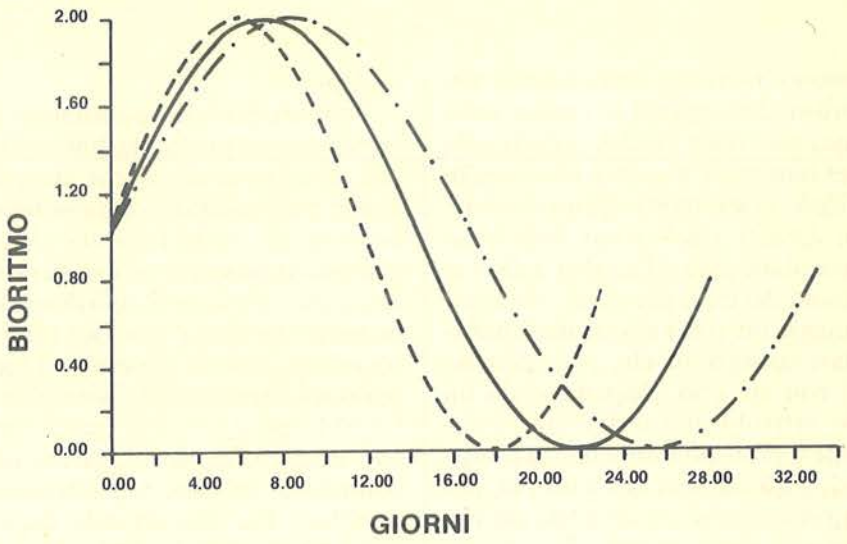


Figura 1. *Vediamo rappresentati i tre cicli: i valori in ordinata non hanno riscontro pratico, mentre in ascissa sono rappresentati i giorni. I cicli sono riconoscibili dalla lunghezza e dai simboli sovrapposti.*

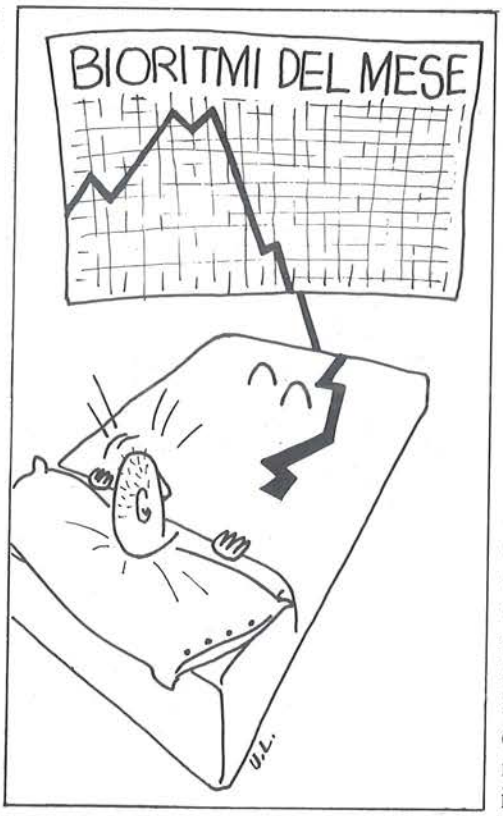
Seguito programma Bioritmi.

```
7,9,12,14,17,19,21,22,23,24,24,2
3,22,21,19,17,14
890 FOR I=1 TO PR
900 READ Y(I)
910 CALL HCHAR(Y(I),I+2,42)
920 NEXT I
930 CALL HCHAR(Y(R)-1,R+2,150)
940 CALL SOUND(90,700,2)
950 CALL KEY(0,KEY,STATUS)
960 IF STATUS=0 THEN 950
970 FOR I=1 TO 28
980 CALL HCHAR(Y(I),I+2,32)
990 NEXT I
1000 CALL HCHAR(14,1,32,312)
1010 CALL HCHAR(13,1,140,32)
1020 CALL HCHAR(Y(R)-1,R+2,32)
1030 FOR I=1 TO LEN(FS$)
1040 P=ASC(SEG$(FS$,I,1))
1050 CALL HCHAR(1,3+I,P)
1060 NEXT I
1070 CALL SCREEN(14)
1080 MN=17
1090 MX=6
```

```
1100 FX=12
1110 PR=23
1120 GOSUB 1290
1130 GOSUB 1650
1140 V$=K$
1150 GOSUB 1530
1160 DATA 13,10,7,5,4,3,3,4,5,7,
10,13,16,19,21,22,23,23,22,21,19
,16,13
1170 FOR I=1 TO 23
1180 READ X(I)
1190 CALL HCHAR(X(I),I+4,42)
1200 NEXT I
1210 CALL HCHAR(X(R)-1,R+4,150)
1220 CALL SOUND(90,700,2)
1230 CALL KEY(0,KEY,STATUS)
1240 IF STATUS=0 THEN 1230
1250 CALL CLEAR
1260 INPUT "vuoi analizzare un'a
ltra data oun altro soggetto?":R
S$
1270 IF RS$="SI" THEN 100
1280 END
1290 REM  CALCOLO MIN,MAX,FLEX
1300 R=RESTO(PR)
```

il più negativo. Questi 3 punti individuano sulla curva delle zone con caratteristiche comuni: nel primo quarto del ciclo riconosciamo la zona più favorevole, caratterizzata da un'ascesa positiva verso il massimo: dovrebbe essere questo il periodo a noi più propizio. Infatti, raggiunto e superato il massimo, pur rimanendo in fase positiva, il bioritmo inizia a calare.

Arrivato al punto critico di flesso, si continua nella discesa; è questo il periodo più brutto. Infatti, una volta superato il minimo, si risentirà dell'effetto positivo dovuto ad una fase di crescita. Si era detto come i cicli coinvolgessero tre componenti essenziali del nostro ego; è necessario distinguere, soprattutto nel primo, la differenza che esiste fra stato bioritmico e stato di forma. Il fatto che noi siamo malati non necessariamente implica che il nostro bioritmo è basso così come un bioritmo positivo indica solamente che noi



siamo ben disposti per difenderci da agenti esterni, non che siamo in perfetta salute in quel determinato momento. Tenete ben presenti queste considerazioni quando sarete tentati di rinnegare la teoria bioritmica perché il vostro bioritmo è alto e voi siete a letto con la febbre. Il secondo ciclo coinvolge la nostra sfera psichica con tutte la manifestazioni consce ed inconsce del nostro io, in particolare affetti, sensi, emotività; nei giorni *no* mettete in guardia chi vi sta accanto. L'ultimo ciclo, quello intellettivo, non merita particolari considerazioni, vi invito solo a calcolarlo in vista di interrogazioni o esami. Se crederete a quanto scritto e farete girare il programma, state anche attenti a non farvi suggestionare attribuendo al vostro bioritmo la responsabilità di eventi casuali. Detto questo non resta altro da fare che invitarvi a digitare il programma sul vostro TI99/4A, utilizzando il TI-BASIC.

```
1310 IF R>MX THEN 1340
1320 MAX=MX-R
1330 GOTO 1350
1340 MAX=PR-(R-MX)
1350 IF R>MN THEN 1380
1360 MIN=MN-R
1370 GOTO 1390
1380 MIN=PR-(R-MN)
1390 IF R<FX THEN 1420
1400 FLEX=PR-R
1410 GOTO 1430
1420 FLEX=FX-R
1430 RETURN
1440 REM  CALCOLO JD
1450 IF A>=0 THEN 1470
1460 A=A+1
1470 K=INT((M-3)/12)
1480 J=INT(((M-2)-(K*12))*30.59)
1490 JD=INT((A+K+4712)*365.25)+J
+G+29
1500 IF JD<2299170 THEN 1520
1510 JD=JD-INT(.75*INT((A+100+K)
/100))+2
1520 RETURN
```

```
1530 REM  SUB.DESCR.
1540 W=15
1550 FOR I=1 TO 135 STEP 15
1560 C$=SEG$(V$,I,15)
1570 FOR J=1 TO LEN(C$)
1580 V=ASC(SEG$(C$,J,1))
1590 CALL HCHAR(W,1+J,V)
1600 NEXT J
1610 W=W+1
1620 NEXT I
1630 RETURN
1640 REM  SUBR.CARATTERIZZA
1650 MIN$=STR$(MIN)
1660 MAX$=STR$(MAX)
1670 FLEX$=STR$(FLEX)
1680 FOR I=1 TO 2
1690 D=ASC(SEG$(MIN$,I,1))
1700 E=ASC(SEG$(MAX$,I,1))
1710 F=ASC(SEG$(FLEX$,I,1))
1720 CALL HCHAR(3,24+I,D)
1730 CALL HCHAR(5,24+I,E)
1740 CALL HCHAR(7,24+I,F)
1750 NEXT I
1760 RETURN
```

Descrizione del listato

L'idea di fondo del programma si basa sul fatto che è possibile individuare lo stato bioritmico calcolando i giorni che si sono vissuti e poi eseguendo l'operazione: (giorni vissuti Mod giorni ciclo). I giorni vissuti saranno calcolati facendo la differenza dei giorni giuliani corrispondenti alla nascita ed alla data in esame; la funzione Mod che dà il resto della divisione intera non è disponibile nella biblioteca del TI99 per cui verrà simulata senza che ciò induca particolari complicazioni. Vediamo ora più dettagliatamente come si comporta il programma. Nelle prime righe viene assegnato il colore bianco a tutti i caratteri e viene cambiato il colore del video da verde in blu, vengono inoltre definiti i caratteri freccia e linea secondo le modalità ben note all'utente del Texas. Le linee seguenti devono contenere le didascalie che appariranno sul video a commento di ogni ciclo; dovranno essere fatte in modo da poter essere troncate correttamente ogni 15 caratteri. Queste, che sono state omesse nel listato, potranno al massimo essere lunghe come una linea di testo e serviranno per commento e per contorno ad ogni ciclo visualizzato. Se il giorno del ciclo intellettivo è il trentaduesimo la freccia non appare perché il video, come noto, è lungo solo 32 caratteri. La routine che viene eseguita dopo ogni input serve per il calcolo del giorno giuliano. Per ulteriori chiarimenti sul funzionamento di detta routine si veda **Personal Software** n.7. Anche il calcolo delle distanze dai punti salienti del ciclo viene demandato ad una routine alla quale passiamo i parametri su cui operare tramite gli assegnamenti che precedono la sua chiamata. Per i meno esperti si fa notare che il ciclo 420-450 serve a fare apparire una stringa in una qualsiasi parte del video senza effettuare un deprecabile scroll verso l'alto. Il ciclo seguente stampa sul video le altre informazioni necessarie ad una corretta comprensione dell'elaborato.

Successivamente si ricorre ad una subroutine per scrivere i valori delle distanze dai punti salienti del ciclo. I valori che appaiono nell'istruzione Data costituiscono le ordinate della curva che verrà disegnata sul video tramite la stampa di asterischi. Il ciclo che segue provvede a stampare correttamente detta curva a cavallo della linea delle ascisse che sarà già apparsa. Infine, quando si passa ad analizzare un nuovo ciclo, vengono sostituite le informazioni riguardanti il precedente ed il processo fin qui descritto viene ripetuto fino all'esaurimento dei cicli da esaminare. ■

E' IN EDICOLA

Bit,
la prima rivista europea
di personal computer,
software, accessori,
la più prestigiosa
e più diffusa in Italia

con tutta la competenza del

GRUPPO
EDITORIALE
JACKSON

solo sproteggere il programma si può usare un metodo più veloce.
Vediamo cosa accade se proteggiamo un programma con una parola di lunghezza inferiore a sette caratteri, ad esempio:

PASS "AAA"

Nei suddetti indirizzi di memoria troveremo la sequenza 81, 81,81, 0, 85, 86, 87. Ciò che ci interessa è osservare che, in questo caso, viene posto uno zero per indicare che i numeri corrispondenti alla password sono finiti (gli ultimi tre numeri riguardano la password che avevamo usato prima). Allora, andando a mettere uno zero nel primo byte di questa sequenza, faremo credere al computer che la password sia di zero caratteri!

Quindi, qualunque sia la password segreta, si può sproteggere il programma semplicemente digitando:

POKE 59561,0
PASS " "

Naturalmente il lavoro del pirata non si ferma qui, perché, ora che abbiamo scoperto un trucco, chi vorrà mantenere segreti i propri programmi ne escogiterà uno ancora più difficile!

Sub Copy

di Sergio Borsani

Parliamo di stampante e soprattutto di come ottenere una hard-copy di ciò che appare sul video in un dato istante. Intendiamoci, se lo schermo presenta solo simboli appartenenti al normale set di caratteri la cosa non presenta alcuna difficoltà, basta leggere il contenuto dello schermo con l'istruzione Call Gchar e stampare i corrispondenti caratteri ASCII. Ben più interessante ed in prospettiva più ricco di applicazioni è il poter riprodurre su carta anche i caratteri definiti dall'utente.
Ahimè, mi sono sempre rammaricato del fatto che il nostro TI Extended BASIC non possedesse i comandi grafici Pset, Line, Circle, Paint, Draw, ecc. ma, con un po' di pazienza e una bella sequenza di Call Char qualcosa si può fare pur con la limitazione dello scarso numero di caratteri ridefinibili. Tuttavia, una volta creati disegni ad alta risoluzione sorge il problema di come stamparli. La stampante a matrice di punti della Texas altro non è che una Epson MX-80 e possiede notevoli capacità grafiche anche se non sono molto diffusi programmi di utilità in grado di sfruttarle pienamente. La routine che qui presentiamo permette di ottenere una esatta copia del video o di una parte di esso, con tutti i caratteri grafici eventualmente presenti. Ovviamente non è assolutamente conveniente usarla se mancano caratteri ridefiniti in quanto con essa il video viene scandito pixel per pixel con i tempi, non certo brevi, che l'operazione comporta. La routine è stata organizzata come sottoprogramma in modo che si possa agganciare ad un normale programma, senza la preoccupazione di usare alcune variabili comuni. Il sottoprogramma, al quale è stato dato il nome di Sub Copy, potrà essere richiamato in particolari momenti alla pressione di un tasto di funzione.

Per sfruttare pienamente le capacità grafiche della stampante occorre mettere nella posizione Off il dip switch SW2-1. È questo un interruttore interno alla stampante, collocato sul quadro dell'interfaccia seriale. L'accesso all'interruttore è semplicissimo. Svitate le quattro viti che tengono fermo il coperchio e tolto quest'ultimo con cautela, si presenta in modo evidente il quadro dell'interfaccia seriale nella posizione indicata nella figura a pag. 101 del manuale. Sul quadro sono presenti due serie di piccoli interruttori, un gruppo a quattro poli ed uno a otto poli. L'interruttore SW2-1 è il primo della serie di quattro ed è tarato dalla casa nella posizione On. In tal modo nella testina di stampa sono attivati sette fili per colonna mentre, ponendo lo stesso interruttore nella posizione Off sono attivati otto fili per colonna. La ragione di tutto ciò è evidente: in un testo c'è sempre una fila di pixel inutilizzata che separa una riga dall'altra ed i caratteri, pur essendo definiti in una matrice di 8 x 8 punti, ne utilizzano al massmo sette in altezza. Il discorso cambia se i caratteri vengono ridefiniti per creare della grafica . In tal caso vengono utilizzati tutti i pixel e conseguentemente nella testina di stampa avremo bisogno di otto fili. Per inciso, c'è un programma di utilità che permette lo stesso risultato senza modificare il dip switch SW2-1, ma è più complicato, di più lenta esecuzione e non è giustificato, data la facilità di accesso all'interruttore interno.
Veniamo più dettagliatamente al sottoprogramma Sub Copy. Come tutti i consimili va posto al termine del

Sub Copy

Listato 1. *Il sottoprogramma Sub Copy.*

```
1000 SUB COPY(R1,R2,C1,C2)
1010 IF R2<R1 OR C2<C1 THEN SUBEXIT
1020 IF R1<1 OR R2>24 OR C1<1 OR C2>32 T
HEN SUBEXIT
1030 OPEN #4:"RS232.CR.DA=8" :: PRINT #4
:CHR$(27);CHR$(65);CHR$(8);
1040 FOR R=R1 TO R2 :: FOR C=C1 TO C2 ::
 CALL GCHAR(R,C,N):: CALL CHARPAT(N,W$)
1050 PRINT #4:CHR$(27);"K";CHR$(8);CHR$(
0);
1060 FOR K=3 TO 0 STEP -1 :: Q=0 :: FOR
E=1 TO 15 STEP 2
1070 D=ASC(SEG$(W$,E,1)):: IF D<60 THEN
D=D-48 ELSE D=D-55
1080 D=D AND 2^K :: IF D THEN Q=Q+2^(7-I
NT(E/2))
1090 NEXT E :: PRINT #4:CHR$(Q);
1100 NEXT K
1110 FOR K=3 TO 0 STEP -1 :: Q=0 :: FOR
E=2 TO 16 STEP 2
1120 D=ASC(SEG$(W$,E,1)):: IF D<60 THEN
D=D-48 ELSE D=D-55
1130 D=D AND 2^K :: IF D THEN Q=Q+2^(8-I
NT(E/2))
1140 NEXT E :: PRINT #4:CHR$(Q);
1150 NEXT K
1160 NEXT C :: PRINT #4:CHR$(13);CHR$(10
)
1170 NEXT R
1180 PRINT #4:CHR$(27);CHR$(65);CHR$(12)
;:: CLOSE #4
1190 SUBEND
```

Listato 2. *Il sottoprogramma Sub Copy con immessi i dati.*

```
100 REM       PROVA
110 REM  SOTTOPROGRAMMA
120 REM *** SUB COPY ***
130 REM
140 DATA FFFFFFFF07070707,FDFDFDFD81818181
150 DATA E0E0E0E0E0E0E0EF,0F1F3F7C7B787C3F
160 DATA C0E0F1FB7B7BFBF9,7EFFFFE7C3C3E7FF
170 DATA 0000B0C0C0C1C1C3,307060E1C2C48992
180 DATA 3E42822262A22222,0304040408090912
190 DATA E0101090884848 24,0707070707070707
200 DATA 8181818181818181,EFEFE0E0E0E0E0E0
210 DATA 1F0F00787F3F1F0F,F8787BFBFBF1E0C0
220 DATA FF7B03C7FFFFFF7E,C3C7C6CECC9C1818
230 DATA 274040407F000000,E3010101E322223E
240 DATA 1213202020474470,24E4020202F1111F
250 FOR J=100 TO 121 :: READ W$
260 CALL CHAR(J,W$):: NEXT J
270 CALL CLEAR
280 FOR J=10 TO 20 :: CALL HCHAR(12,J,90+J):: NEXT J
290 FOR J=10 TO 20 :: CALL HCHAR(13,J,101+J):: NEXT J
300 CALL GOPY(12,13,10,20)
310 END
1000 SUB COPY(R1,R2,C1,C2)
     .......
1220 SUBEND
```

programma utente, dopo l'istruzione End. Viene chiamato con una Call, rispettando una precisa sintassi come se si trattasse di una nuova istruzione. La forma da usare è: Call Copy (R1, R2, C1, C2), dove R1, R2, C1 e C2 sono i numeri delle righe e delle colonne che delimitano la parte di video da copiare. Se i valori indicati non rientrano nell'intervallo ammesso il sottoprogramma Copy non viene eseguito e la sua chiamata non ha alcun effetto.

La linea 1030 del listato apre il file #4 per comunicare con la stampante e fissa a 8 l'interlinea in modo che ogni

Figura 1. *L'hard-copy su stampante.*

linea di stampa sia a diretto contatto con quella che la precede.

La linea 1040 inizia il ciclo di lettura della porzione di video da copiare. Successivamente viene dichiarato l'invio di 8 byte per la stampa in bit-map a densità normale (linea 1050). Nel computer il pattern dei caratteri è definito in codice esadecimale ed occupa 8 byte corrispondenti a 8 righe di pixel, ordinate dall'alto verso il basso; invece, un analogo carattere sulla stampante, viene definito da 8 valori decimali corrispondenti a 8 colonne ordinate da sinistra a destra. Pertanto l'istruzione 1070 ricava uno ad uno i codici esadecimali e li trasforma in valori decimali mentre la 1080 ricava il numero Q corrispondente ad una fila verticale di otto punti nel modo grafico in bit image. Il ciclo viene compiuto due volte per i due semibyte di sinistra e di destra e le linee 1110-1150 svolgono una funzione analoga a quella fino ad ora considerata. Interessante è notare l'uso dell'operatore logico And alle linee 1080 e 1130; poiché ognuno dei 16 codici esadecimali che definiscono un carattere corrispondono a 4 pixel, l'operazione booleana And mette in evidenza la presenza di uno dei 4 punti alla volta, al variare del valore di 2^ K.

Per illustrare il funzionamento del programma abbiamo riportato il listato 2. In esso vengono definiti alcuni caratteri per creare la scritta TI99/4A in formato gigante. Dopo le istruzioni che fanno apparire la scritta sullo schermo si nota, alla linea 300, la chiamata del sottoprogramma Call Copy (...). Come è stato detto, il programma principale deve terminare con l'istruzione End alla quale farà seguito il sottoprogramma Sub Copy (listato 1).

I possessori del sistema a dischi troveranno molto utile agganciare con un Merge questa utility a molti programmi già in loro possesso. ■

PERSONAL SOFTWARE

Text-Editor per TI99/4A

Formattiamo i nostri testi con il TI99

di *Sergio Borsani*

L'uso del personal come elaboratore di testi è ormai così diffuso che è diventata una tendenza comune, soprattutto nel campo dei portatili, corredare il computer di un Text-Editor residente su ROM, eventualmente accompagnato da altri programmi di data base, wailing list e grafica commerciale.
Il TI99/4A dispone, a tale proposito, del modulo SSS Writer TI, forse difficilmente reperibile in commercio al giorno d'oggi. Il programma comprende molte funzioni avanzate le quali tuttavia non compensano, a nostro avviso, la necessità di dover leggere il testo di 80 colonne attraverso finestre contigue ed in parte sovrapposte di 40 caratteri ciascuna. Come giustamente è scritto in *Things to do with your TI99/4A computer*, J. Willis, M. Miller & D. LaMont Johnson, New American Library, dopo aver usato il Writer TI vi sembrerà di aver assistito ad una partita di tennis da bordo campo!

Molti usano l'Editor/Assembler come word-processor, ma è chiaro che oltre a non eliminare il problema delle finestre, esso comporta la perdita di molte importanti funzioni come l'inserimento di una parola con il riaggiustamento dell'intero capoverso. In altre parole un inserimento è possibile solo nell'ambito di una linea di testo e comporta la perdita della parte finale della linea stessa. Inoltre con l'opzione Print dell'Editor/ Assembler non è possibile controllare il formato della stampa.

Dopo questa premessa non sembrerà più tanto infondata la proposta di un nuovo word-processor per il TI99/4A.

Scritto in Extended BASIC, il programma si attiene alle seguenti caratteristiche di base: normale output su video con 28 caratteri per riga, con la possibilità di usare un nuovo set di caratteri stile IBM, con un vero corsivo; testo in forma di capoversi; controllo del formato di stampa tramite codici inseriti tra un capoverso e l'altro; giustificazione del margine destro di stampa con suddivisione in sillabe per il ritorno a capo riga (!); mantenimento di tutte le principali funzioni dell'Editor/ Assembler, quali Delete, Move, Copy, Insert, che permettono la cancellazione e lo spostamento di grossi blocchi di testo compreso il merge e la fusione di due testi diversi.
Il programma occupa circa 12 Kbyte di memoria e necessita del sistema a dischi, l'espansione di memoria e la stampante. Esso permette di gestire un testo fino ad un massimo di 300 righe per 28 caratteri, corrispondente a 15 videate.

La carta di identità di questo text-editor è abbastanza eloquente e permette di collocarlo su un piano di competitività, se non di superiorità, rispetto ad altri programmi analoghi quali *Direct Writer II* della Dynamic Data & Devices, *Word-Processing* della Futura Software Inc. o *Textiger*, tutti reperibili negli U.S.A.
Il programma è diviso in due blocchi; il primo, denominato DSK1. Textnewset, contiene la definizione del nuovo set di caratteri; il secondo DSK1. Texteditor, è il programma vero e proprio. Il primo si allaccia automaticamente al secondo, il quale può anche essere eseguito in modo autonomo, nel qual caso però si dispone solo del normale set di caratteri.

Il menu fornisce 6 opzioni, nell'ordine: Load, Edit, Save, Print, Purge e Files. Con Load si carica in memoria un file precedentemente registrato. Edit permette di creare un testo o editare quello presente in memoria. Parleremo più avanti di tutte le funzioni di editing. Save memorizza il testo presente in memoria nel formato Sequential, Display, Variable 80. Ogni record contiene due righe di testo di 28 caratteri ed il file così ottenuto può essere caricato anche con l'Editor Assembler. Sia per il comando Load che per il Save, va specificato il nome del file secondo le norme consuete; ad esempio è valido il nome DSK1.Myfile. Il comando Print permette la stampa del testo attualmente in memoria. E' necessario specificare il nome del dispositivo, ad esempio RS-232, e successivamente il margine sinistro di stampa e

Text-Editor per TI99/4A

la larghezza del testo. I valori di default sono 10 e 60, nel qual modo vengono lasciate 10 colonne di margine a destra e a sinistra del foglio. Ulteriori codici di controllo della stampa possono venir inseriti nel testo così da ottenere la stampa ingrandita, condensata, enfatizzata, doppia ed inoltre il salto riga o l'avanzamento del modulo, ecc. Il comando Purge annulla completamente il testo in memoria, dopo la conferma da parte dell'utente. Infine, con Files si ottiene la lista dei file Dis/Var 80 con la possibilità di cancellarli, uno alla volta, dalla directory.

Veniamo ora all'editing. Selezionato il comando Edit, si passa allo schermo contenente il testo. Questo occupa le 20 righe centrali. Le prime due righe, nello schermo 1, contengono il nome del file, o la scritta Newfile se si inizia un nuovo testo; negli schermi successivi, nella stessa posizione, vengono mostrate le ultime due righe della videata precedente, in modo da consentire continuità nella scrittura. Le ultime due righe in basso sono riservate alle funzioni Delete, Move, Copy, ecc.

Durante la scrittura si hanno a disposizione due cursori, uno "vuoto" ed uno "pieno" lampeggiante, corrispondenti a due diversi modi di editing. Si passa da un modo all'altro semplicemente premendo il tasto Enter. Il cursore lampeggiante indica che ci si trova nell'editing di linea; si può scrivere, spostare il cursore a destra e a sinistra, inserire e cancellare caratteri, ma solo nell'ambito di una riga. Premendo Enter compare il cursore vuoto a capo riga ed in questo modo si ha a disposizione un editing di schermo, potendo spostare il cursore in ogni parte del video, cancellare o inserire righe, inserire o cancellare caratteri con il riaggiustamento dell'intero capoverso, accedere alle funzioni Move, Copy, Insert, ecc.

Durante la scrittura è necessario premere il tasto Enter al termine di ogni riga. Se si desidera usufruire delle funzioni generali di editing basta un leggero tocco e si ottiene il cursore vuoto a capo riga, se invece si desidera continuare la scrittura basta mantenere la pressione sul tasto Enter e si ottiene il cursore "pieno" lampeggiante, sempre a capo riga. Per scrivere bisogna sempre essere in presenza del cursore lampeggiante.

Anche se si deve premere Enter al termine di ogni riga, ogni capoverso o l'intero testo, sono virtualmente considerati dal programma come un'unica lunga riga. Pertanto, quando si va a capo non si devono rispettare le comuni regole grammaticali per la suddivisione in sillabe, né usare i trattini di unione. Ad esempio, se si sta scrivendo la parola computer e si termina la riga con "comp", si continuerà a capo con il seguito "uter". Analogamente, terminando una riga con una parola completa, si porrà uno spazio all'inizio della riga successiva, esattamente come se si proseguisse sulla stessa riga. Ci penserà poi il programma a formattare il testo nel modo specificato dall'utente.

Il testo può essere suddiviso in capoversi di lunghezza variabile. Al limite tutto il testo può essere costituito da un unico capoverso, ma in tal caso l'esecuzione di alcune funzioni risulterebbe più lenta. I capoversi devono venir separati da una riga vuota con una barra obliqua inversa nella prima colonna (\ = Fctn z).

Dopo aver ripetuto con il cursore nero lampeggiante si dispone delle funzioni di editing di linea, cioè Delete Char, Insert Char, Erase, e dei comandi di controllo cursore a destra e a sinistra, ma solo nell'ambito di una riga, vediamo le funzioni generali di editing.

A queste si accede in presenza del cursore vuoto, il quale compare premendo il tasto Enter. Sono abilitati i seguenti tasti di funzione: Delete Char = Fctn 1; Insert Char = Fctn 2; Delete Line = Fctn 3; Next Screen = Fctn 5; Roll Down = Fctn 6; Aid = Fctn 7; Insert Line = Fctn 8 ed Escape = Fctn 9. Attenzione perchè, mentre è disabilitato il comando Break, tranne che in presenza del menu, è sempre attivo il comando Quit = Fctn + che corrisponde ad un Reset di sistema.

Il significato delle funzioni è evidente, tuttavia è opportuno fare qualche precisazione. Del Char e Ins Char hanno valore per l'intero capoverso. Delete Line cancella la riga dove è posto il cursore e riaggiusta il testo. Next Screen, mancando le finestre di testo laterali, ha lo stesso significato

Listato 1. *Il nuovo insieme di caratteri.*

```
100 REM     NUOVO SET
110 REM    DI CARATTERI
120 REM ****************
130 CALL CLEAR :: CALL CHAR(33,"00181818
18180018"):: CALL CHAR(34,"006C6C6C"):: 
CALL CHAR(37,"0073566C18366ACE")
140 !@P-
150 CALL CHAR(40,"0008103030301008"):: C
ALL CHAR(41,"0010080C0C0C0810"):: CALL C
HAR(43,"000018187E7E1818")
155 CALL CHAR(45,"000000007E7E0000"):: C
ALL CHAR(38,"0020505020D6CC76")
160 CALL CHAR(47,"0002060C18306040"):: C
ALL CHAR(48,"003C66666E76663C"):: CALL C
HAR(49,"00183818181818 3C")
170 CALL CHAR(50,"003C66060C18307E"):: C
ALL CHAR(51,"003C66060C06663C"):: CALL C
HAR(52,"000C1C2C4C7E0C0C")
180 CALL CHAR(53,"007E607C0606663C"):: C
ALL CHAR(54,"003C60607C66663C"):: CALL C
HAR(55,"007E460C18306060")
190 CALL CHAR(56,"003C66663C66663C"):: C
ALL CHAR(57,"003C66663E06663C"):: CALL C
HAR(60,"000C18306030180C")
200 CALL CHAR(62,"0030180C060C1830"):: C
ALL CHAR(63,"003C66660C180018"):: CALL C
HAR(65,"00183C66667E6666")
210 CALL CHAR(66,"00FC66667C6666FC"):: C
ALL CHAR(67,"003C66606060663C"):: CALL C
HAR(68,"00FC6666666666FC")
220 CALL CHAR(69,"00FE6260786062FE"):: C
ALL CHAR(70,"00FE6260786060F0"):: CALL C
HAR(71,"003C66606E66663C")
230 CALL CHAR(72,"006666667E666666"):: C
ALL CHAR(73,"003C18181818183C"):: CALL C
HAR(74,"003C181818185830")
240 CALL CHAR(75,"0062666C78786C66"):: C
ALL CHAR(76,"00F06060606062FE"):: CALL C
HAR(77,"00C6EEFED6C6C6C6")
250 CALL CHAR(78,"00C6E6F6DECEC6C6"):: C
ALL CHAR(79,"003C66666666663C"):: CALL C
HAR(80,"00FC6666667C60F0")
260 CALL CHAR(81,"003C6666666A643A"):: C
ALL CHAR(82,"00FC66667C6666F3"):: CALL C
HAR(83,"003C66603C06663C")
270 CALL CHAR(84,"007E5A181818183C"):: C
ALL CHAR(85,"006666666666663C"):: CALL C
HAR(86,"00666666243C1818")
280 CALL CHAR(87,"00C6D6D6D6D67C28"):: C
ALL CHAR(88,"0066663C183C6666"):: CALL C
HAR(89,"0066663C18181818")
290 CALL CHAR(90,"007E460C1830627E"):: C
ALL CHAR(91,"003C30303030303C"):: CALL C
HAR(93,"003C0C0C0C0C0C3C")
300 CALL CHAR(97,"00000380C7CCC76"):: C
ALL CHAR(98,"0060607C6666667C"):: CALL C
HAR(99,"0000003C6660663C")
310 CALL CHAR(100,"0006063E6666663E"):: 
CALL CHAR(101,"0000003C667E603C"):: CALL
 CHAR(102,"001C303078303030")
320 CALL CHAR(103,"0000003A663E463C"):: 
CALL CHAR(104,"0060607C66666666"):: CALL
 CHAR(105,"0018003818181 83C")
330 CALL CHAR(106,"0018003818185830"):: 
CALL CHAR(107,"006060666C786C66"):: CALL
 CHAR(108,"0038181818181 83C")
340 CALL CHAR(109,"000000ECB6B6B6B6"):: 
CALL CHAR(110,"0000005C66666666"):: CALL
 CHAR(111,"0000003C6666663C")
350 CALL CHAR(112,"0000007C66667C60"):: 
CALL CHAR(113,"0000003E66663E06"):: CALL
 CHAR(114,"0000006E76606060")
360 CALL CHAR(115,"0000003C603C067C"):: 
CALL CHAR(116,"0030307C3030301C"):: CALL
 CHAR(117,"0000006666666 63C")
370 CALL CHAR(118,"0000006666663C18"):: 
CALL CHAR(119,"000000C6D6D67C28"):: CALL
 CHAR(120,"000000663C183C66")
380 CALL CHAR(121,"000000663C185830"):: 
CALL CHAR(122,"0000007E4C18327E"):: CALL
 CHAR(123,"001C303060303 01C")
390 CALL CHAR(125,"00380C0C060C0C38"):: 
CALL CHAR(128,"FF8181818181 81FF")
400 RUN "DSK1.TEXTEDITOR"
```

del comando Roll Up dell'Editor/Assembler, cioè fa vedere lo schermo successivo. Roll Down, invece, riporta allo schermo precedente. Aid permette di ottenere la numerazione delle righe ed il numero dello schermo attuale. Insert Line inserisce una linea vuota in corrispondenza del cursore che può essere spostato in ogni posizione del video usando i tasti con le frecce. Infine, Escape mette a disposizione ulteriori comandi generali di editing che appaiono indicati nelle righe inferiori del video. Essi sono nell'ordine: Edit, Move, Copy, Delete, Insert ed ancora Escape, tutte richiamabili con la sola lettera iniziale tranne Escape ottenibile nel solito modo con Fctn 9.

Edit permette di tornare nel modo edit.

Move consente di rimaneggiare il testo spostando un certo numero di righe in un'altra posizione. Dopo aver osservato i numeri di riga con il tasto Aid, si devono indicare i numeri della prima e dell'ultima linea del segmento da-spostare ed il numero di riga oltre la quale va posto, tutti separati da virgole. (Move) 10,15,35 sposta la parte di testo dalla riga 10 alla 15 dopo la linea 35.

Copy svolge una funzione simile a Move, con la differenza che la parte di testo indicata non viene rimossa, ma solo copiata.

Delete permette la cancellazione di un blocco di testo specificando la prima e l'ultima riga della parte da cancellare.

Insert consente la fusione di due file, ovvero di quello attualmente in memoria con un altro memorizzato in precedenza su disco. E' necessario specificare il nome del file del quale si richiede il merging ed il numero di riga dopo il quale dev'essere inserito. Con questo comando è possibile, ad esempio, richiamare l'intestazione di una lettera o un indirizzo tratto

Listato 2. *Il programma Text-Editor.*

```
100 REM  TEXT EDITOR 28/C
110 REM ----------------------
120 REM by Sergio Borsani.
130 ON BREAK NEXT :: ON WARNING NEXT
140 DIM RIGA$(300):: NUMRIGHE=1 :: F$="
      *** (EOF) ***          " :: RIGA$(1)
=F$ :: FINA$="NEW"
150 CALL CLEAR :: CALL SCREEN(13):: FOR
J=1 TO 12 :: CALL COLOR(J,16,13):: NEXT
J :: CALL COLOR(14,16,1)
160 CALL CHAR(128,"FF8181818181FF"):: 
CALL CHAR(129,"FF"):: CALL CHAR(130,"000
0000000000FF")
170 CALL CHAR(131,RPT$("01",8)):: CALL C
HAR(132,RPT$("80",8)):: CALL CHAR(136,"3
C4299A1A199423C"):: GOTO 250
180 CALL HCHAR :: CALL VCHAR :: SC$ :: N
 :: RECORD$ :: X :: SCHERMO :: FINE :: C
ALL SPRITE :: RI :: CO
190 R$ :: NU :: CALL KEY :: K :: S :: CA
LL DELSPRITE :: CALL LOCATE :: Z$ :: KK
:: W$ :: F1 :: T
200 R1 :: R2 :: R3 :: BLOCCO :: CTR :: Q
 :: PX :: FNW$ :: FINA$ :: MS :: LF :: U
R :: PNT :: RIG$ :: DEV$
210 PX1 :: PX2 :: RIGA0$ :: LR0 :: ST$ :
: STS :: L1$ :: L2$ :: L3$ :: L4$
220 CALL VOCALE :: CALL MARGINE :: V ::
DIFF :: COD :: LOOP :: FN$ :: W
230 FLAG :: DN$ :: Y :: PY
240 !@P-
250 CALL CLEAR :: DISPLAY AT(2,4):"T E X
 T    E D I T O R" :: DISPLAY AT(4,2):"c
opyright "&CHR$(136);"1984, S.Borsani"
260 CALL HCHAR(1,3,129,28):: CALL HCHAR(
5,3,130,28):: CALL VCHAR(1,2,131,5):: CA
LL VCHAR(1,31,132,5)
270 DISPLAY AT(8,4):"1  LOAD" :: DISPLAY
 AT(10,4):"2  EDIT" :: DISPLAY AT(12,4):
"3  SAVE"
280 DISPLAY AT(14,4):"4  PRINT" :: DISPL
AY AT(16,4):"5  PURGE"
290 DISPLAY AT(18,4):"6  FILES" :: DISPL
AY AT(20,10):"scelta?"
300 CALL VCHAR(7,4,32,12):: ON BREAK STO
P :: ACCEPT AT(20,18)SIZE(1)VALIDATE("12
3456"):SC$
310 ON ERROR 2490 :: ON BREAK NEXT :: N=
VAL(SC$):: CALL HCHAR(6+2*N,4,62):: ON N
 GOTO 320,390,1270,1310,2020,2040
320 GOSUB 2200 :: ON ERROR 2220
330 NUMRIGHE=1 :: OPEN #1:FINA$,SEQUENTI
AL,INPUT ,DISPLAY ,VARIABLE 80
340 IF EOF(1)THEN 380
350 LINPUT #1:RECORD$ :: X=POS(RECORD$,F
$,1)
360 FOR J=0 TO 1 :: RIGA$(NUMRIGHE)=SEG$
(RECORD$,28*J+1,28):: IF X=28*J+1 THEN 3
80
370 NUMRIGHE=NUMRIGHE+1 :: NEXT J :: GOT
O 340
380 CLOSE #1 :: GOSUB 2390 :: GOTO 300
390 CALL CLEAR :: SCHERMO=1 :: IF NUMRIG
HE>20 THEN FINE=20 ELSE FINE=NUMRIGHE
400 GOSUB 2410 :: FOR J=1 TO FINE :: DIS
PLAY AT(J+2,1):RIGA$(J):: NEXT J
410 CALL SPRITE(#1,128,2,17,17)
420 RI=3 :: CO=1 :: GOTO 480
430 GOSUB 2250 :: ACCEPT AT(RI,CO)SIZE(-
(29-CO)):R$ :: R$=R$&RPT$(" ",29-CO-LEN(
R$))
440 IF CO=1 THEN RIGA$(NU)=R$ ELSE RIGA$
(NU)=SEG$(RIGA$(NU),1,CO-1)&R$
450 CO=1 :: IF RI<22 THEN RI=RI+1 :: GOT
O 470
460 RI=3 :: SCHERMO=SCHERMO+1 :: CALL CL
EAR :: GOSUB 2330 :: GOSUB 2320 :: IF NU
=NUMRIGHE THEN DISPLAY AT(3,1):F$
470 CALL SPRITE(#1,128,2,(RI-1)*8+1,17)
480 CALL KEY(5,K,S):: IF S=0 THEN 480
490 IF K=13 THEN CALL DELSPRITE(#1):: GO
TO 430
500 IF K=1 THEN 630
510 IF K=3 THEN GOSUB 2320 :: Y=CO :: GO
TO 790
520 IF K=4 THEN 720
530 IF K=6 THEN 780
540 IF K=7 THEN 690
550 IF K=8 AND CO>1 THEN CO=CO-1 :: GOTO
 620
560 IF K=9 AND CO<28 THEN CO=CO+1 :: GOT
O 620
570 IF K=10 AND RI<22 THEN RI=RI+1 :: GO
SUB 2250 :: GOTO 620
580 IF K=11 AND RI>3 THEN RI=RI-1 :: GOT
O 620
590 IF K=12 AND SCHERMO>1 THEN 820
600 IF K=14 THEN 830
610 IF K=15 THEN 840 ELSE 480
620 CALL LOCATE(#1,(RI-1)*8+1,(CO+1)*8+1
):: GOTO 480
630 FOR J=3 TO 11 :: CALL HCHAR(J,2,46+J
):: NEXT J
640 FOR J=13 TO 21 :: CALL HCHAR(J,2,36+
J):: NEXT J
650 Z$="SCHERMO "&STR$(SCHERMO)&"  RIGHE
 "&STR$((SCHERMO-1)*20+1)&"-"&STR$(SCHER
MO*20)
660 DISPLAY AT(24,1):Z$
670 CALL KEY(0,KK,S):: IF S=0 THEN 670
680 CALL VCHAR(3,2,32,20):: CALL HCHAR(2
4,3,32,28):: GOTO 480
690 GOSUB 2320 :: IF NU=NUMRIGHE OR NUMR
IGHE=1 THEN 480
700 FOR J=NU TO NUMRIGHE-1 :: RIGA$(J)=R
IGA$(J+1):: NEXT J :: NUMRIGHE=NUMRIGHE-
1 :: GOSUB 2330
710 CALL HCHAR(J,3,32,28):: GOTO 480
720 GOSUB 2320 :: W$=SEG$(RIGA$(NU),28,1
):: IF CO=1 THEN RIGA$(NU)=CHR$(32)&SEG$
(RIGA$(NU),1,27):: GOTO 740
730 RIGA$(NU)=SEG$(RIGA$(NU),1,CO-1)&CHR
$(32)&SEG$(RIGA$(NU),CO,28-CO)
740 NU=NU+1 :: IF NU=NUMRIGHE OR SEG$(RI
GA$(NU),1,1)="\" THEN 750 ELSE 770
750 IF W$<>CHR$(32)THEN GOSUB 2370 :: RI
```

Seguito listato 2.

```
GA$(NU)=W$&RPT$(" ",27)
760 GOSUB 2330 :: GOTO 480
770 Z$=SEG$(RIGA$(NU),28,1):: RIGA$(NU)=
W$&SEG$(RIGA$(NU),1,27):: W$=Z$ :: GOTO
740
780 GOSUB 2320 :: GOSUB 2370 :: GOSUB 23
30 :: GOTO 480
790 IF NU+1=NUMRIGHE OR SEG$(RIGA$(NU+1)
,1,1)="\" THEN W$=CHR$(32):: F1=1 ELSE W
$=SEG$(RIGA$(NU+1),1,1)
800 IF Y=1 THEN RIGA$(NU)=SEG$(RIGA$(NU)
,2,27)&W$ ELSE RIGA$(NU)=SEG$(RIGA$(NU),
1,Y-1)&SEG$(RIGA$(NU),Y+1,28-Y)&W$
810 IF F1=1 THEN F1=0 :: GOSUB 2330 :: G
OTO 480 ELSE NU=NU+1 :: Y=1 :: GOTO 790
820 SCHERMO=SCHERMO-1 :: CO=1 :: RI=3 ::
 CALL CLEAR :: GOSUB 2330 :: GOTO 480
830 IF SCHERMO*20>=NUMRIGHE THEN 480 ELS
E SCHERMO=SCHERMO+1 :: RI=3 :: CO=1 :: C
ALL CLEAR :: GOSUB 2330 :: GOTO 620
840 CALL DELSPRITE(#1):: DISPLAY AT(23,1
):"  <E><M><C><D><I> <ESCAPE>":" " :: FO
R T=1 TO 50 :: NEXT T
850 CALL KEY(5,K,S):: IF S=0 THEN 850
860 IF K=15 THEN CALL DELSPRITE(#1):: GO
TO 250
870 IF K=69 OR K=101 THEN GOSUB 2390 ::
GOTO 620
880 IF K=77 OR K=109 THEN 920
890 IF K=67 OR K=99 THEN 1050
900 IF K=68 OR K=100 THEN 1110
910 IF K=73 OR K=105 THEN 1150 ELSE 850
920 DISPLAY AT(23,1):"MOVE) DA,A,DOPO LA
 RIGA" :: ON ERROR 2470 :: ACCEPT AT(24,
1):R$ :: GOSUB 2500
930 IF R3<R1 OR R3>R2 AND R1<=R2 THEN 96
0
940 DISPLAY AT(24,1):"* ERRORE * PRESS <
ENTER>."
950 CALL KEY(0,K,S):: IF S=0 OR K<>13 TH
EN 950 ELSE 840
960 BLOCCO=R2-R1+1 :: IF NUMRIGHE+BLOCCO
<300 THEN 990
970 GOSUB 2390 :: DISPLAY AT(24,1):"* ME
MORY FULL * PRESS ENTER."
980 CALL KEY(0,K,S):: IF S=0 OR K<>13 TH
EN 980 ELSE 840
990 CTR=NUMRIGHE :: FOR J=R1 TO R2 :: CT
R=CTR+1 :: RIGA$(CTR)=RIGA$(J):: NEXT J
1000 IF R3>R2 THEN 1020
1010 FOR J=R2 TO R3+BLOCCO+1 STEP -1 ::
RIGA$(J)=RIGA$(J-BLOCCO):: NEXT J :: GOT
O 1030
1020 FOR J=R1 TO R3-BLOCCO :: RIGA$(J)=R
IGA$(J+BLOCCO):: NEXT J
1030 IF R3<R1 THEN Q=0 ELSE Q=BLOCCO
1040 FOR J=1 TO BLOCCO :: RIGA$(R3-Q+J)=
RIGA$(NUMRIGHE+J):: NEXT J :: GOTO 390
1050 DISPLAY AT(23,1):"COPY) DA,A,DOPO L
A RIGA" :: ON ERROR 2470 :: ACCEPT AT(24
,1):R$ :: GOSUB 2500
1060 IF R3<R1 OR R3>=R2 AND R1<=R2 THEN
1070 ELSE 940
1070 BLOCCO=R2-R1+1 :: IF NUMRIGHE+BLOCC
O>300 THEN 970
1080 CTR=0 :: FOR J=NUMRIGHE+BLOCCO TO R
3+BLOCCO+1 STEP -1 :: RIGA$(J)=RIGA$(NUM
RIGHE-CTR):: CTR=CTR+1 :: NEXT J
1090 NUMRIGHE=NUMRIGHE+BLOCCO :: IF R3>R
2 THEN Q=0
1100 CTR=0 :: FOR J=R3+1 TO R3+BLOCCO ::
 RIGA$(J)=RIGA$(R1+Q+CTR):: CTR=CTR+1 ::
 NEXT J :: GOTO 390
1110 DISPLAY AT(23,1):"DELETE) DA RIGA,A
 RIGA" :: ON ERROR 2470 :: ACCEPT AT(24,
1):R$ :: GOSUB 2500
1120 IF R1>R2 OR R1<1 OR R2>NUMRIGHE-1 T
HEN 940
1130 Q=R2-R1+1 :: CTR=0 :: FOR J=R1 TO N
UMRIGHE-Q :: CTR=CTR+1 :: RIGA$(J)=RIGA$
(R2+CTR):: NEXT J
1140 NUMRIGHE=NUMRIGHE-Q :: GOTO 390
1150 DISPLAY AT(23,1):"INSERT) FILE NAME
,AFTER LINE" :: ON ERROR 2470 :: DISPLAY
 AT(24,1):"DSK1."
1160 ACCEPT AT(24,1)SIZE(-28):R$ :: PX=P
OS(R$,",",1):: IF PX=0 OR PX>16 THEN 940
1170 FNW$=SEG$(R$,1,PX-1):: R3=VAL(SEG$(
R$,PX+1,LEN(R$)-PX)):: BLOCCO=0
1180 OPEN #1:FNW$,SEQUENTIAL,INPUT ,DISP
LAY ,VARIABLE 80
1190 IF EOF(1)THEN 1210
1200 LINPUT #1:RECORD$ :: X=POS(RECORD$,
F$,1):: IF X=0 THEN BLOCCO=BLOCCO+2 :: G
OTO 1190 ELSE BLOCCO=BLOCCO+INT(X/28)
1210 IF BLOCCO+NUMRIGHE>300 THEN CLOSE #
1 :: GOTO 970 ELSE RESTORE #1 :: CTR=0
1220 FOR J=NUMRIGHE+BLOCCO TO R3+BLOCCO+
1 STEP -1 :: RIGA$(J)=RIGA$(J-BLOCCO)::
NEXT J
1230 IF EOF(1)THEN 1260 ELSE LINPUT #1:R
ECORD$ :: X=POS(RECORD$,F$,1)
1240 FOR J=0 TO 1 :: IF X=28*J+1 THEN 12
60 ELSE CTR=CTR+1 :: RIGA$(R3+CTR)=SEG$(
RECORD$,28*J+1,28)
1250 NEXT J :: GOTO 1230
1260 CLOSE #1 :: NUMRIGHE=NUMRIGHE+BLOCC
O :: GOTO 390
1270 GOSUB 2200 :: J=1 :: ON ERROR 2220
1280 OPEN #1:FINA$,SEQUENTIAL,OUTPUT,DIS
PLAY ,VARIABLE 80
1290 RECORD$=RIGA$(J)&RIGA$(J+1):: PRINT
 #1:RECORD$ :: IF J+1<NUMRIGHE THEN J=J+
2 :: GOTO 1290
1300 CLOSE #1 :: GOSUB 2390 :: GOTO 300
1310 GOSUB 2190 :: IF SC$="N" OR SC$="n"
 THEN GOSUB 2390 :: GOTO 300
1320 GOSUB 2390 :: DISPLAY AT(23,1):"dev
ice name:" :: DISPLAY AT(24,1):"RS232"
1330 ACCEPT AT(24,1)SIZE(-28):DEV$ :: CA
LL CLEAR
1340 ON ERROR 2430 :: OPEN #4:DEV$ :: DI
SPLAY AT(18,1):"* STAMPA *":" ":"MARGINE
 SINISTRO = 10"
1350 ACCEPT AT(20,20)SIZE(-2)VALIDATE(DI
GIT):MS :: IF MS<1 OR MS>80 THEN 1370
```

Seguito listato 2.

```
1360 DISPLAY AT(21,1):"NUMERO COLONNE =
60" :: ACCEPT AT(21,18)SIZE(-2)VALIDATE(
DIGIT):LF :: IF LF>1 AND LF<80-MS+1 THEN
 1390
1370 DISPLAY AT(23)BEEP:"* ERRORE * FORM
ATO STAMPA":"* PRESS <ENTER> TO CONTINUE
"
1380 CALL KEY(0,K,S):: IF S=0 OR K<>13 T
HEN 1380 ELSE GOSUB 2390 :: GOTO 1350
1390 DISPLAY AT(23,1):"PER INTERROMPERE
LA STAMPA  PREMI (FCTN)(9)=ESCAPE."
1400 UR=NUMRIGHE-1 :: J=1 :: W=MS :: ON
ERROR 2440
1410 FNT=LF :: RIG$=""
1420 CALL KEY(5,K,S):: IF K=15 THEN CLOS
E #4 :: GOTO 250
1430 IF SEG$(RIGA$(J),1,1)="\" THEN PX1=
1 :: GOTO 1930
1440 RIGAO$=RIGA$(J):: LRO=LEN(RIGAO$)
1450 IF PNT<LRO THEN 1490
1460 RIG$=RIG$&RIGAO$ :: PNT=PNT-LRO ::
J=J+1
1470 IF PNT=0 THEN 1500
1480 IF J>UR THEN 1500 ELSE 1420
1490 RIG$=RIG$&SEG$(RIGAO$,1,PNT)
1500 ST$=RIG$ :: STS=0
1510 REM      CONTROLLO
1520 REM     A CAPO RIGA
1530 IF LEN(ST$)<LF THEN 1820
1540 L1$=SEG$(ST$,LF-1,1)
1550 L2$=SEG$(ST$,LF,1)
1560 IF PNT<>0 THEN 1580
1570 IF J>UR THEN 1820 ELSE RIGAO$=RIGA$
(J)
1580 L3$=SEG$(RIGAO$,PNT+1,1):: L4$=SEG$
(RIGAO$,PNT+2,1)
1590 IF L3$=CHR$(32)THEN STS=STS-1 :: GO
TO 1800
1600 IF L2$=L3$ OR L3$=L4$ THEN 1750
1610 IF L1$=L2$ THEN 1790
1620 IF L1$="C" AND L2$="Q" THEN 1790
1630 IF L1$="c" AND L2$="q" THEN 1790
1640 IF L2$=CHR$(32)THEN 1750
1650 PX=POS(",.;:-/+=)%!?'`"&CHR$(34),L2
$,1):: IF PX<>0 THEN 1800
1660 PX=POS(",.;:-/+=)%!?'`"&CHR$(34),L3
$,1):: IF PX<>0 THEN 1750
1670 PX=POS("LRMNlrmn",L1$,1):: IF PX=0
THEN 1690
1680 CALL VOCALE(L2$,V):: IF V=0 THEN 17
90
1690 CALL VOCALE(L1$,V):: IF V=0 THEN 17
50
1700 CALL VOCALE(L2$,V):: IF V=1 THEN 17
50
1710 CALL VOCALE(L3$,V):: IF V=1 THEN 17
90
1720 IF L3$="R" OR L3$="r" THEN 1790
1730 IF L2$<>"S" AND L2$<>"s" THEN 1750
1740 IF L2$<>L3$ THEN 1790
1750 STS=STS+1 :: IF STS>10 OR STS>LF-1
THEN ST$=RIG$ :: GOTO 1820
1760 L4$=L3$ :: L3$=L2$ :: L2$=L1$ :: L1
$=SEG$(ST$,LF-1-STS,1)
1770 ST$=SEG$(ST$,1,LF-STS):: GOTO 1590
1780 GOTO 1820
1790 ST$=SEG$(ST$,1,LEN(ST$)-1)&CHR$(45)
:: STS=STS+1
1800 DIFF=LF-LEN(ST$)
1810 IF DIFF>0 THEN CALL MARGINE(ST$,DIF
F)
1820 PRINT #4:RPT$(" ",W);:: IF KK=1 THE
N W=MS
1830 PRINT #4:ST$
1840 IF J>UR THEN 1920
1850 PNT=PNT-STS
1860 IF PNT=28 THEN J=J+1 :: GOTO 1410
1870 IF PNT<1 THEN 1890
1880 RIGAO$=SEG$(RIGAO$,PNT+1,LRO-PNT)::
 LRO=LEN(RIGAO$):: GOTO 1910
1890 IF PNT=0 THEN 1410
1900 J=J-1 :: RIGAO$=SEG$(RIGA$(J),LRO+P
NT+1,-PNT):: LRO=LEN(RIGAO$)
1910 PNT=LF :: RIG$="" :: GOTO 1450
1920 CLOSE #4 :: GOTO 250
1930 IF LEN(RIG$)=0 THEN 1970
1940 FOR Q=1 TO LEN(RIG$):: IF ASC(SEG$(
RIG$,Q,1))<>32 THEN PRINT #4:RPT$(" ",W)
;RIG$ :: GOTO 1960
1950 NEXT Q :: GOTO 1970
1960 IF KK=1 THEN W=MS
1970 PX2=POS(RIGA$(J),"\",PX1+1):: IF PX
2=0 THEN J=J+1 :: GOTO 1410
1980 COD=VAL(SEG$(RIGA$(J),PX1+1,PX2-PX1
-1)):: IF COD=14 THEN W=W/2 :: KK=1 ELSE
 KK=0
1990 IF COD=15 THEN W=INT(W*33/20)+1
2000 IF COD=18 THEN W=MS
2010 PRINT #4:CHR$(COD);:: PX1=PX2 :: GO
TO 1970
2020 GOSUB 2190 :: IF SC$="Y" OR SC$="y"
 THEN NUMRIGHE=1 :: FINA$="NEW" :: RIGA$
(1)=F$
2030 GOSUB 2390 :: GOTO 300
2040 CALL CLEAR :: ON ERROR 2530 :: OPEN
 #3:"DSK1.",INPUT ,RELATIVE,INTERNAL ::
CTR=0 :: T=-7
2050 INPUT #3:DN$,J,J,K :: PRINT "DISKNA
ME=";DN$:"available=";K;"used";J-K
2060 PRINT :: FOR LOOP=1 TO 127 :: INPUT
 #3:FN$,Y,J,K :: IF LEN(FN$)=0 THEN 2120
2070 IF ABS(Y)=2 THEN PRINT FN$;TAB(19+T
);STR$(J),:: CTR=CTR+1 :: T=-T
2080 IF CTR<36 THEN 2110
2090 PRINT :" PRESS <ENTER> TO CONTINUE"
2100 CALL KEY(0,K,S):: IF S=0 THEN 2100
ELSE PRINT :: CTR=0
2110 NEXT LOOP
2120 CLOSE #3 :: IF T=+7 THEN PRINT
2130 PRINT :"DELETE FILE? (Y/N) N" :: AC
CEPT AT(23,20)SIZE(-1)VALIDATE("YNyn"):R
$
2140 IF R$="N" OR R$="n" THEN 250
2150 FLAG=1 :: ON ERROR 2220 :: GOSUB 23
90 :: GOSUB 2200 :: GOSUB 2190
2160 IF SC$="N" AND SC$="n" THEN 250
```

Seguito listato 2.

```
2170 OPEN #5:FINA$,SEQUENTIAL,INPUT ,DIS
PLAY ,VARIABLE 80 :: CLOSE #5:DELETE
2180 FLAG=0 :: FINA$=FNW$ :: GOTO 250
2190 DISPLAY AT(24,6)BEEP:"SEI SICURO? (
Y/N) N" :: ACCEPT AT(24,24)SIZE(-1)VALID
ATE("YNyn"):SC$ :: RETURN
2200 FNW$=FINA$ :: DISPLAY AT(23,1):"fil
e name: DSK1." :: ACCEPT AT(23,12)SIZE(-
15):FINA$
2210 RETURN
2220 DISPLAY AT(23,1):"* ERRORE * FILE N
OT PRESENT" :: DISPLAY AT(24,1):"(PRESS
<ENTER> TO CONTINUE)"
2230 CALL KEY(0,K,S):: IF S=0 OR K<>13 T
HEN 2230 ELSE GOSUB 2390 :: FINA$=FNW$
2240 IF FLAG=1 THEN FLAG=0 :: RETURN 250
 ELSE RETURN 300
2250 GOSUB 2320 :: IF NU<NUMRIGHE THEN R
ETURN
2260 IF NUMRIGHE<300 THEN 2290
2270 DISPLAY AT(23,1)BEEP:"* WARNING * O
UT OF DIMENSION" :: DISPLAY AT(24,1):"pr
emi  ENTER per continuare"
2280 CALL KEY(0,K,S):: IF S=0 OR K<>13 T
HEN 2280 ELSE CALL HCHAR(23,3,32,60):: R
I=RI-1 :: RETURN
2290 RIGA$(NUMRIGHE)=RPT$(" ",28):: NUMR
IGHE=NUMRIGHE+1 :: RIGA$(NUMRIGHE)=F$
2300 DISPLAY AT(RI,1):RIGA$(NUMRIGHE-1):
: IF RI<22 THEN DISPLAY AT(RI+1,1):F$
2310 RETURN
2320 NU=(SCHERMO-1)*20+RI-2 :: RETURN
2330 IF SCHERMO*20<=NUMRIGHE THEN FINE=2
2 ELSE FINE=NUMRIGHE+2-(SCHERMO-1)*20
2340 GOSUB 2320 :: Y=0 :: FOR J=RI TO FI
NE :: DISPLAY AT(J,1):RIGA$(NU+Y):: Y=Y+
1 :: NEXT J
2350 IF SCHERMO=1 THEN GOSUB 2410 :: RET
URN ELSE NU=(SCHERMO-1)*20 :: DISPLAY AT
(1,1):RIGA$(NU-1):: DISPLAY AT(2,1):RIGA
$(NU)
2360 RETURN
2370 NUMRIGHE=NUMRIGHE+1 :: FOR J=NUMRIG
HE TO NU+1 STEP -1 :: RIGA$(J)=RIGA$(J-1
):: NEXT J
2380 RIGA$(NU)=RPT$(" ",28):: RETURN
2390 CALL HCHAR(23,3,32,60):: RETURN
2400 CALL HCHAR(1,3,32,60):: RETURN
2410 IF FINA$="NEW" THEN W$="edit NEW FI
LE" ELSE W$="edit "&FINA$
2420 DISPLAY AT(1,1):W$ :: DISPLAY AT(2,
1):RPT$("-",28):: RETURN
2430 DISPLAY AT(23,1):"* ERRORE * STAMPA
NTE" :: GOTO 2450
2440 DISPLAY AT(23,1):"* ERRORE * CODICI
 CONTROLLO"
2450 DISPLAY AT(24,1):"(PREMI ENTER PER
CONTINUARE)"
2460 CALL KEY(0,K,S):: IF S=0 OR K<>13 T
HEN 2460 ELSE CLOSE #4 :: RETURN 250
2470 DISPLAY AT(24,1):"* ERRORE * PRESS
<ENTER>."
2480 CALL KEY(0,K,S):: IF S=0 OR K<>13 T
HEN 2480 ELSE RETURN 840
2490 RETURN 300
2500 PX=POS(R$,",",1):: IF PX=0 THEN R1=
VAL(R$):: R2,R3=0 :: RETURN
2510 R1=VAL(SEG$(R$,1,PX-1)):: PY=POS(R$
,",",PX+1):: IF PY=0 THEN R2=VAL(SEG$(R$
,PX+1,LEN(R$)-PX)):: R3=0 :: RETURN
2520 R2=VAL(SEG$(R$,PX+1,PY-PX-1)):: R3=
VAL(SEG$(R$,PY+1,LEN(R$)-PY)):: RETURN
2530 PRINT " * ERROR * DISK NOT PRESENT"
:"PRESS <ENTER> TO CONTINUE."
2540 CALL KEY(0,K,S):: IF S=0 OR K<>13 T
HEN 2540
2550 ON ERROR 2560 :: CLOSE #3
2560 RETURN 250
2570 !@P+
2580 END
2590 SUB VOCALE(LX$,V)
2600 V=POS("AEIOUaeiou",LX$,1):: IF V<>0
 THEN V=1
2610 SUBEND
2620 SUB MARGINE(RX$,W)
2630 LS=LEN(RX$):: CTR=0
2640 FOR J=1 TO LS-1 :: L=ASC(SEG$(RX$,L
S-J,1))
2650 IF L=32 THEN RX$=SEG$(RX$,1,LS-J)&"
 "&SEG$(RX$,LS-J+1,J+CTR):: CTR=CTR+1
2660 IF CTR=W THEN SUBEXIT
2670 NEXT J :: IF CTR<W THEN W=W-CTR ::
GOTO 2630
2680 SUBEND
```

da una mailing list.

I comandi Copy e Insert possono portare ad una saturazione della memoria disponibile nel qual caso compare il messaggio ★ MEMORY FULL ★ PRESS ENTER, che rende invalido il comando impedendo la sua esecuzione senza che il programma debba subire un blocco.

Escape fa tornare al menu principale.

Consideriamo ora il modo in cui si possono inserire nel testo codici di controllo per la stampa. Si è detto che i capoversi in cui viene suddiviso il testo devono venir separati da una riga con una barra obliqua inversa (Fctn z) nella prima posizione. Ebbene, dopo tale segno si possono inserire i codici decimali, riportati nel manuale della stampante, ognuno seguito da una barra obliqua inversa. Ad esempio, \ 10 \ produce il salto di una riga; \ 15 \ causa la stampa di caratteri condensati; e ancora \ 27 \ 69 \ ha come effetto la stampa nel modo enfatizzato, ecc.

In pratica si possono usare tutti i codici previsti dal manuale; l'unica limitazione sta nel fatto che che non è possibile inserire nella stessa riga diverse modalità di stampa. Tuttavia si può sempre ottenere l'intestazione di una lettera con caratteri diversi, ad esempio condensati, evidenziare un titolo con caratteri enfatizzati, ecc. Resta inteso che il programma riesce a tenere allineato il margine sinistro anche in presenza

di caratteri ingranditi o condensati.

Siamo convinti che questo Text-Editor, pur non dotato di tutte le funzioni di Writer TI, ne elimina il difetto dovuto alla presenza di finestre laterali ed offre una buona leggibilità accompagnata dalla facilità d'uso. A chi poi adopera attualmente l'Editor/ Assembler come elaboratore di testi, dà innegabili vantaggi per la sua capacità di inserimento e cancellazione di caratteri con il riaggiustamento dell'intero capoverso e di controllo del formato stampa. In ogni caso il fiore all'occhiello resta la sua prerogativa di giustificazione del margine destro con la suddivisione delle parole in sillabe secondo la grammatica italiana. Una caratteristica davvero poco comune anche nei word-processor più sofisticati.

DIFF	Numero di space da inserire in una riga di stampa per giustificare il margine destro.
COD	Numero di codice per il controllo della stampa.
FLAG	Variabile flag usata in una subroutine richiamata da un'istruzione On Error; in base al suo valore il programma viene rimandato all'inizio del menu o all'accept del menu stesso.
FN$	Nome di un file letto dal catalogo del dischetto.
DN$	Nome del dischetto.

Principali variabili del programma

RIGA$() – Righe di testo di 28 caratteri ciascuna.
NUMRIGHE – Numero di righe delle quali è composto il testo in memoria.
FINA$ – Nome del file in memoria.
SC$ – Scelta dell'utente tra le operazioni del menu.
RECORD$ – Variabile di lavoro per la lettura o la scrittura di un record da o su disco. Ogni record contiene due righe di testo.

Figura 1. *Menu principale che presenta le stesse funzioni dell'Editor/Assembler con l'aggiunta dell'opzione Files con la quale si possono leggere e cancellare i file presenti sul dischetto.*

Figura 2. *Premendo Edit si può creare un nuovo testo o editarne uno già presente in memoria. Il cursore "vuoto" indica che ci si trova nel modo comandi.*

SCHERMO – Numero dello schermo sul quale è presente una parte di testo; corrisponde al numero della pagina video.
FINE – Variabile di lavoro per mostrare la prima pagina di testo.
RI,CO – Numero di riga e di colonna sul quale si trova il cursore a forma di quadrato vuoto che si ha nel modo funzioni.
R$ – Variabile che accetta momentaneamente una riga scritta dall'utente.
NU – Numero di riga che si sta editando.
Z$,W$ – Variabili di lavoro usate prevalentemente per ridefinire le righe dopo un inserimento o la cancellazione di un carattere.

F1 – Variabile flag che indica la fine del testo o di un capoverso durante la ridefinizione delle righe in seguito alla cancellazione o all'inserimento di un carattere.
R1,R2,R3 – Memorizzano rispettivamente la prima e l'ultima riga di un blocco di testo e il numero di riga oltre il quale va collocato. Le variabili sono usate con le funzioni: Move, Copy, Insert, Delete.
BLOCCO – Lunghezza, espressa in numero di righe, del blocco di testo da inserire o da copiare.
CTR – Variabile con funzione di contatore.
PX – Posizioni in cui si trovano le virgole che separano i numeri di riga specificati dopo una funzione Move, Copy, ecc.
FNW$ – Variabile usata per memorizzare momentaneamente il nome di un file.
DEV$ – Nome dell'interfaccia per la stampante.

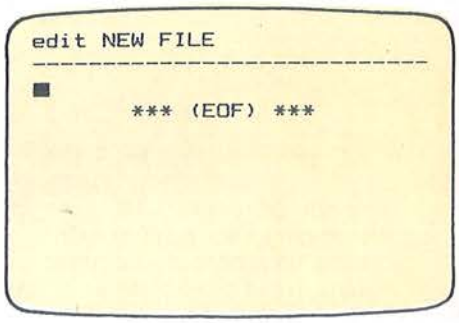


Figura 3. *Premendo il tasto Enter compare il cursore nero lampeggiante e si inizia la scrittura.*

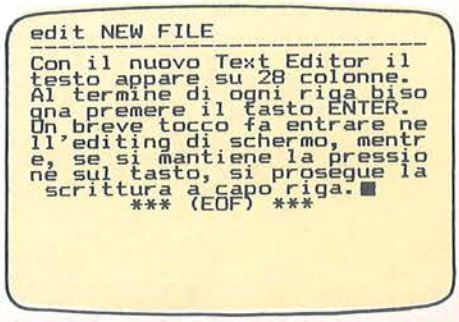


Figura 4. *Con il nuovo Text-Editor non si è costretti a leggere il testo attraverso finestre contigue.*

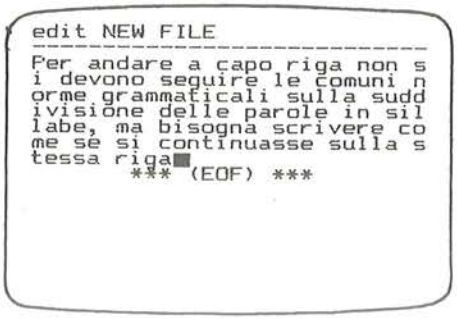


Figura 5. *Ogni capoverso viene considerato dal programma come un'unica lunga riga di testo. La formattazione avviene al momento della stampa.*

Figura 6. *La barra obliqua inversa separa un capoverso dall'altro.*

MS – Numero di colonne lasciate libere oltre il margine sinistro nel testo stampato.
LF – larghezza del formato di stampa.
UR – numero di righe da stampare.
PNT – Variabile di lavoro per formattare le righe per la stampa.
RIG$ – Riga di LF caratteri prima del controllo per andare a capo riga nel modo grammaticalmente corretto.
PX1,PX2 – Posizioni delle barre oblique inverse nelle stringhe contenenti i codici di controllo per la stampa.
RIGA0$ – Variabile di lavoro usata per formattare le righe per la stampa.
LR0 – Lunghezza di una riga nel formato video.
ST$ – Riga nel formato stampa.
STS – Numero di caratteri di cui deve essere ridotta la riga nel formato di stampa per poter andare a capo correttamente. La lunghezza LF viene poi ripristinata distribuendo un uguale numero di space all'interno della riga.
L1$,L2$ – Penultimo ed ultimo carattere di una riga nel formato stampa.
L3$,L4$ – Primo e secondo carattere della riga successiva a quella che dev'essere stampata. La loro conoscenza è indispensabile per poter andare a capo in modo corretto.
V – Variabile flag che indica la presenza di una vocale.

Principali routine

100-240 – Inizializzazione di tutte le variabili e dei sottoprogrammi per il pre-scan.
250-310 – Titolo e menu. La linea 300 riabilita il Break (Fctn 4) in modo da poter fermare il programma.

Figura 7. *Esempio dell'uso di un codice di controllo della stampa.*

Figura 8. *Premendo Escape (Fctn 9) nel modo comandi si accede a funzioni che permettono il trasferimento di grossi blocchi di testo così come il merge di un file da disco con la fusione di due testi.*

Figura 9. *La larghezza del formato stampa viene definita dall'utente al momento stesso della stampa.*

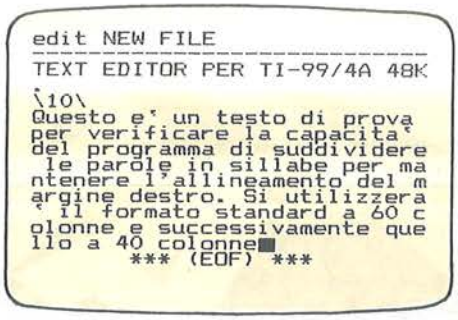


Figura 10. *Testo di prova.*

Nelle altre fasi ciò è possibile solo con il comando Quit (Fctn +).
320-380 – Caricamento di un programma da disco (Load).
390-420 – Visualizzazione della prima pagina video nel modo Edit.
430-460 – Viene accettata una riga di stampa.
470-610 – Studio della tastiera per accettare i comandi di editing.
620 – Controllo cursore.
630-680 – Esecuzione comando Aid.
690-710 – Esecuzione comando Delete Line.
720-770 – Esecuzione comando Insert Char.
780 – Esecuzione comando Insert Line.
790-810 – Esecuzione comando Delete Char.
820 – Esecuzione comando Roll Down.
830 – Esecuzione comando Next Screen.
840-910 – Esecuzione comando Escape e attesa per ulteriori comandi di editing.
920-1040 – Comando Move.

```
TEXT EDITOR PER TI-99/4A 48K.

Questo e' un testo di prova per verifi-
care la capacita' del programma di sud-
dividere le parole in sillabe per mante-
nere l'allineamento del margine destro.
Si utilizzera' il formato standard a 60
colonne e successivamente quello a 40
colonne.
```

Figura 11. *Stampa del testo di prova su 40 colonne.*

```
TEXT EDITOR PER TI-99/4A 48K.

Questo e' un testo di prova per verificare la capacita' del
programma di suddividere le parole in sillabe per mantenere
l'allineamento del margine destro. Si utilizzera' il formato
standard a 60 colonne e successivamente quello a 40 colonne.
```

Figura 12. *Stampa del testo di prova su 60 colonne.*

1050-1100 – Comando Copy.
1110-1140 – Comando Delete.
1150-1260 – Comando Insert.
1270-1300 – Memorizzazione di un file su disco (Save).
1310-1410 – Preliminari per la stampa.
1420-1500 – Formattazione della riga di stampa.
1510-1810 – Routine per andare a capo riga secondo le regole della grammatica italiana.
1820-1830 – Stampa di una riga.
1840-1910 – Routine per inizializzare una nuova riga nel formato stampa.
1920 – Termine della stampa.
1930-2010 – Routine per la lettura e l'esecuzione dei comandi per il controllo della stampa.
2020-2030 – Esecuzione del comando Purge.
2040-2180 – Lettura dei file presenti sul disco ed eventuale loro cancellazione.
2190 – Subroutine per attendere una conferma.
2200-2210 – Richiesta del nome del file.
2220-2240 – Gestione degli errori durante il Load o quando si esegue la cancellazione di un file.
2250-2280 – Controllo della lunghezza del testo.
2290-2310 – Subroutine per predisporre una riga vuota per la scrittura e per far avanzare la riga che indica la fine del testo (Eof).
2320 – Calcolo del numero di riga dove si trova il cursore.
2330-2360 – Mostra una pagina di testo sul video.
2370-2380 – Subroutine per inserire una riga vuota nel testo in memoria.
2390-2400 – Subroutine per cancellare rispettivamente le ultime e le prime due righe del video.
2410-2420 – Scrittura del nome del file nella prima riga della prima pagina del testo.
2430 – Messaggio d'errore.
2440-2460 – Gestione degli errori durante la stampa.
2470-2480 – Controllo degli errori in fase di editing.
2490 – Subroutine per rimandare il programma all'Accept del menu principale in caso di errore.
2500-2520 – Determinazione della posizione delle virgole che separano i numeri di riga nei comandi Move, Copy, Delete e Insert.
2530-2560 – Subroutine per la gestione degli errori nelle operazioni I/O su disco.
2570 – Ripristina il pre-scan prima dei sottoprogrammi BASIC.
2580 – Fine del programma principale.
2590-2610 – Sottoprogramma per la ricerca di una vocale.
2620-2680 – Sottoprogramma per inserire uniformemente degli space in una riga di stampa in modo che risulti allineato il margine destro. ■

PERSONAL SOFTWARE

Caccia al tredici

Anche il vostro Texas TI99 può vincere al Totocalcio

di *Carlo Aceti*

Caccia al tredici è un programma che aiuta nella stesura della schedina del Totocalcio, soprattutto se non si è degli esperti in materia. Prima di passare al listato, spenderemo due parole per spiegare cosa fare dopo il Run. Anzitutto, dato che il programma per funzionare ha bisogno di alcuni dati, bisognerà dapprima procurarsi un giornale specializzato, dal quale si potranno attingere le informazioni desiderate.
Dopo il Run vedremo l'intestazione, quindi apparirà il titolo, infine verranno fatte le prime domande, di carattere generale, sulla schedina in questione.
Una di queste domande, "Quante doppie max vuoi?", necessita forse di un chiarimento.

Con essa viene richiesto il numero massimo di doppie che desiderate; non è detto però che vi saranno date tutte: ne otterrete tante quante saranno state elaborate dal computer; solo nel caso che esse siano più di quelle richieste, il computer provvederà ad eliminare l'eccedenza.
Se dopo l'ultima domanda, "Tutto OK?", la risposta è affermativa (basta solo un Enter) lo schermo cambierà colore e si passerà alla terza pagina, ossia alle domande relative alle squadre che giocano in casa.
Cioè:

Listato 1. *Il programma Totocalcio.*

```
10 CALL CLEAR :: CALL SCREEN(11)
:: PRINT " CACCIA AL 13": : :
20 PRINT " PROGRAMMA PER TI-99/4
A": : :
30 PRINT " EXT.BASIC ": : :
40 PRINT " REALIZZATO DA ACETI C
ARLO": : :
50 PRINT :: PRINT
60 CALL RITARDO(1000)
70 PRINT "$$$    BUONA FORTUNA
  $$$": : :
80 CALL RITARDO(1000)
90 PRINT :: PRINT :: PRINT
100 CALL TITOLO
110 CALL SCREEN(11)
120 OPTION BASE 1
130 DIM A$(26,2),RIS$(13,1),B$(1
3,1):: A=4 :: K=0 :: AA=4 :: C=4
 :: CC=4 :: Q=1
140 CALL CLEAR
150 DISPLAY AT(3,5):"RISPONDI AL
LE DOMANDE"
160 CALL LINO(5,2,30):: CALL LIN
O(10,2,30):: CALL LINO(14,2,30):
: CALL LINO(18,2,30)
170 DISPLAY AT(6,5):"QUANTE SONO
 LE PAR-" :: DISPLAY AT(7,5):"TI
TE CHE SI GIOCANO" :: DISPLAY AT
(8,5):"IN SERIE A,B,C1? >"
180 ACCEPT AT(8,24)VALIDATE(NUME
RIC)SIZE(2):TT :: ON WARNING NEX
T :: TOT=TT
190 DISPLAY AT(11,5):"QUANTE DOP
PIE" :: DISPLAY AT(12,5):"MASSIM
O VUOI?" :: ACCEPT AT(12,24)VALI
DATE(NUMERIC)SIZE(2):X
200 ON WARNING NEXT
210 DISPLAY AT(15,5):"CHE NUMERO
 DI" :: DISPLAY AT(16,5):"PARTIT
E SIAMO ?" :: ACCEPT AT(16,24)VA
LIDATE(NUMERIC)SIZE(2):GIO :: GI
O1=GIO
220 ON WARNING NEXT
230 GIO=GIO/100
240 DISPLAY AT(20,6):"TUTTO OK<S
/N)?" :: ACCEPT AT(20,20)VALIDAT
E("NS")SIZE(-1):G$ :: CALL CLEAR

250 IF G$="N" OR G$="N" THEN GOT
O 150
260 CALL SCREEN(16)
270 F2$="IN CASA"
280 F3$="TRASF. "
290 F$="VIT. PAR. SCO. DIF."
300 FOR I=1 TO TOT
310 CALL LINO(3,1,32):: CALL LIN
O(19,1,32)
320 DISPLAY AT(1,2):F2$&" "&F$
```

Caccia al tredici

Seguito listato Totocalcio.

```
330 CALL NUM
340 ACCEPT AT(A,4)VALIDATE(UALPH
A)SIZE(5):A$(I,1):: ACCEPT AT(A,
10)VALIDATE(NUMERIC)SIZE(2):V
350 ON WARNING NEXT
360 ACCEPT AT(A,15)VALIDATE(NUME
RIC)SIZE(2):P :: ACCEPT AT(A,20)
VALIDATE(NUMERIC)SIZE(2):S
370 ACCEPT AT(A,25)VALIDATE(NUME
RIC)SIZE(2):D
380 RI=(2*V)+(1*P)-(1.1*S)-(0.3*
D)
390 IF GIO1>(V+P+S)THEN RI=(RI/(
V+P+S))*GIO1
400 ON WARNING NEXT
410 A=A+1 :: IF A=17 THEN CALL H
CHAR(4,1,32,480):: A=4
420 A$(I,2)=STR$(RI)
430 NEXT I
440 TOT=TOT+1
450 IF TT=0 THEN TOT=1
460 FOR I=TOT TO 13
470 DISPLAY AT(1,2):F2$&" "&F$
480 CALL NUM
490 CALL LINO(3,1,32):: CALL LIN
O(19,1,32)
500 ACCEPT AT(A,4)VALIDATE(UALPH
A)SIZE(5):A$(I,1):: ACCEPT AT(A,
10)VALIDATE(NUMERIC)SIZE(2):VV
510 ACCEPT AT(A,15)VALIDATE(NUME
RIC)SIZE(2):PP :: ACCEPT AT(A,20
)VALIDATE(NUMERIC)SIZE(2):SS
520 ACCEPT AT(A,25)VALIDATE(NUME
RIC)SIZE(2):DD
530 A=A+1 :: IF A=17 THEN CALL H
CHAR(4,1,32,480):: A=4
540 ON WARNING NEXT
550 RI2=((VV*1)+(PP*.5)-(SS*.7))
/(DD*.3):: A$(I,2)=STR$(RI2)
560 NEXT I
570 IF TOT<>0 THEN TOT=TOT-1
580 FOR I=14 TO 13+TOT
590 IF TT=0 THEN GOTO 710
600 DISPLAY AT(1,2):F3$&" "&F$
610 CALL NUM
620 ACCEPT AT(A,4)VALIDATE(UALPH
A)SIZE(5):A$(I,1):: ACCEPT AT(A,
10)VALIDATE(NUMERIC)SIZE(2):V1
630 ON WARNING NEXT
640 ACCEPT AT(A,15)VALIDATE(NUME
RIC)SIZE(2):P1 :: ACCEPT AT(A,20
)VALIDATE(NUMERIC)SIZE(2):S1
650 ACCEPT AT(A,25)VALIDATE(NUME
RIC)SIZE(2):D1
660 RI1=(2.1*V1)+(1.5*P1)-(.9*S1
)-(0.3*D1)
670 IF GIO1>(V1+P1+S1)THEN RI1=(
RI1/(V1+P1+S1))*GIO1
680 ON WARNING NEXT
690 A=A+1 :: A$(I,2)=STR$(RI1)
700 NEXT I
710 IF TT=0 THEN TOT=1 ELSE TOT=
TOT+1
720 FOR I=13+TOT TO 26
730 DISPLAY AT(1,2):F3$&" "&F$
740 CALL NUM
750 ACCEPT AT(A,4)VALIDATE(UALPH
A)SIZE(5):A$(I,1):: ACCEPT AT(A,
10)VALIDATE(NUMERIC)SIZE(2):VV
760 ACCEPT AT(A,15)VALIDATE(NUME
RIC)SIZE(2):PP :: ACCEPT AT(A,20
)VALIDATE(NUMERIC)SIZE(2):SS
770 ACCEPT AT(A,25)VALIDATE(NUME
RIC)SIZE(2):DD
780 ON WARNING NEXT
790 RI2=((VV*1)+(PP*.5)-(SS*.7))
/(DD*.3):: A=A+1 :: A$(I,2)=STR$
(RI2)
800 NEXT I
810 CALL CLEAR
820 CALL LINV(4,11,13):: CALL LI
NV(4,19,13)
830 CALL LINO(3,1,32):: CALL LIN
O(18,1,32)
840 CALL TITF
850 FOR I=1 TO 13
860 DISPLAY AT(C,1)SIZE(7):A$(I,
1)
870 C=C+1 :: NEXT I
880 FOR D=14 TO 26
890 DISPLAY AT(CC,10)SIZE(7):A$(
D,1)
900 CC=CC+1 :: NEXT D
910 FOR IN=1 TO 13
920 SC1=VAL(A$(IN,2))
930 SC2=VAL(A$(IN+13,2))
940 GOSUB 1000
950 NEXT IN
960 GOSUB 1220
970 GOSUB 1350
980 CALL KEY(0,T,S):: IF S=0 THE
N 980 :: IF T=67 THEN GOTO 110
990 END
1000 REM CONTEGGIO
1010 IF SC1>SC2 THEN GOTO 1030
1020 IF SC1<SC2 THEN GOTO 1100
1030 IF (SC1-SC2)>=INT((.05+GIO)
```

Seguito listato Totocalcio.

```
*SC2)THEN RI$="2"
1040 IF (SC1-SC2)>=INT((.10+GIO)
*SC2)THEN RI$="2X"
1050 IF (SC1-SC2)>=INT((.19+GIO)
*SC2)THEN RI$="X1"
1060 IF (SC1-SC2)>=INT((.29+GIO)
*SC2)THEN RI$="X"
1070 IF (SC1-SC2)>=INT((.35+GIO)
*SC2)THEN RI$="1X"
1080 IF (SC1-SC2)>=INT((.45+GIO)
*SC2)THEN RI$="1"
1090 GOTO 1160
1100 IF (SC2-SC1)>=INT((.05+GIO)
*SC1)THEN RI$="1"
1110 IF (SC2-SC1)>=INT((.10+GIO)
*SC1)THEN RI$="1X"
1120 IF (SC2-SC1)>=INT((.19+GIO)
*SC1)THEN RI$="X"
1130 IF (SC2-SC1)>=INT((.29+GIO)
*SC1)THEN RI$="X1"
1140 IF (SC2-SC1)>=INT((.35+GIO)
*SC1)THEN RI$="2X"
1150 IF (SC2-SC1)>=INT((.45+GIO)
*SC1)THEN RI$="2"
1160 IF LEN(RI$)=2 THEN K=K+1
1170 IF K>X THEN RI$=SEG$(RI$,1,
1)ELSE RI$=RI$
1180 RIS$(Q,1)=RI$ :: Q=Q+1
1190 DISPLAY AT(AA,20)SIZE(2):RI
$
1200 AA=AA+1
1210 RETURN
1220 REM CASUALITA`
1230 AA=4
1240 FOR I=1 TO 13
1250 RANDOMIZE
1260 S=INT(4*RND)+1
1270 ON S GOSUB 1290,1300,1310,1
320
1280 GOTO 1330
1290 B$(I,1)="1" :: RETURN
1300 B$(I,1)="1X" :: RETURN
1310 B$(I,1)="2" :: RETURN
1320 B$(I,1)="X" :: RETURN
1330 DISPLAY AT(AA,23)SIZE(2):B$
(I,1)
1340 AA=AA+1 :: NEXT I :: RETURN
1350 AA=4
1360 REM MEDIA
1370 RANDOMIZE
1380 FOR I=1 TO 13
1390 MED=INT(2*RND)+1
1400 ON MED GOSUB 1420,1430
1410 GOTO 1440
1420 MED$=RIS$(I,1):: RETURN
1430 MED$=B$(I,1):: RETURN
1440 IF LEN(MED$)=2 THEN W=W+1
1450 IF W>X THEN MED$=SEG$(MED$,
1,1)ELSE MED$=MED$
1460 DISPLAY AT(AA,26)SIZE(2):ME
D$ :: AA=AA+1
1470 NEXT I
1480 RETURN
1490 SUB TITOLO :: CALL CLEAR ::
 CALL COLOR(2,5,5)
1500 A5$=" + ++  +  + +++" :: A6
$="+++  ++++  ++++"
1510 A1$="++++  +  +++" :: A2$="
++++ +++++ +" :: A3$="+  +  +  +
  " :: A4$="+  +  +  +++"
1520 FOR I=1 TO 4 :: DISPLAY AT(
I+7,5):SEG$(A1$,I*3-2,3):: NEXT
I
1530 CALL SOUND(10,1000,4)
1540 FOR I=1 TO 4 :: DISPLAY AT(
I+7,9):SEG$(A2$,I*3-2,3):: NEXT
I
1550 CALL SOUND(10,1000,4)
1560 FOR I=1 TO 4 :: DISPLAY AT(
```

- il nome della squadra;
- le vittorie in casa;
- i pareggi in casa;
- le sconfitte in casa;
- coefficiente di difficoltà (in relazione all'abilità della squadra avversaria).

Quest'ultima domanda costituisce un tocco personale riservato ai più esperti. Potete "penalizzare" una squadra assegnandole un coefficiente di difficoltà, variabile da 0 a 99, nell'affrontare la squadra avversaria.

Dopo le tredici squadre che giocano in casa, automaticamente si passerà a quelle che giocano in trasferta.

Due precisazioni:

- nel listato il nome della squadra non potrà essere lungo più di 5 lettere;
- quando scrivete il nome delle squadre, ricordate che il computer confronterà la prima delle squadre che giocano in casa con la prima di quelle che giocano in trasferta, procedendo in modo analogo per tredici volte. Sarebbe perciò consigliabile seguire l'ordine indicato dalla schedina.

L'ultima pagina è completamente gestita dal nostro TI99.

Vedremo apparire i nomi delle squadre che si affrontano e tre tipi di risultati:

1) logico: ossia il logico risultato scaturito dai dati che sono stati immessi;
2) random: cioè scelta casuale ad opera del computer;
3) misto: dove casualmente vengono mischiati i due tipi di risultati precedenti, nella speranza di poter vincere più facilmente.

Grazie ad un key, il video rimane fermo per tutto il tempo che vogliamo: premendo la C maiuscola torneremo alle prime domande; premendo qualsiasi altro tasto fermeremo il programma.

Ed ora il listato.

Caccia al tredici

Seguito listato Totocalcio.

```
I+7,13):SEG$(A1$,I*3-2,3):: NEXT
 I
1570 CALL SOUND(10,1000,4)
1580 FOR I=1 TO 4 :: DISPLAY AT(
I+7,17):SEG$(A1$,I*3-2,3):: NEXT
 I
1590 CALL SOUND(10,1000,4)
1600 FOR I=1 TO 4 :: DISPLAY AT(
I+7,21):SEG$(A3$,I*3-2,3):: NEXT
 I
1610 CALL SOUND(10,1000,4)
1620 FOR I=1 TO 4 :: DISPLAY AT(
I+7,23):SEG$(A2$,I*3-2,3):: NEXT
 I
1630 CALL SOUND(10,1000,4)
1640 FOR I=1 TO 4 :: DISPLAY AT(
I+13,7):SEG$(A2$,I*3-2,3):: NEXT
 I
1650 CALL SOUND(10,1000,4)
1660 FOR I=1 TO 4 :: DISPLAY AT(
I+13,11):SEG$(A4$,I*3-2,3):: NEX
T I
1670 CALL SOUND(10,1000,4)
1680 FOR I=1 TO 5 :: DISPLAY AT(
I+12,16):SEG$(A5$,I*3-2,3):: NEX
T I
1690 CALL SOUND(10,1000,4)
1700 FOR I=1 TO 5 :: DISPLAY AT(
I+12,21):SEG$(A6$,I*3-2,3):: NEX
T I
1710 CALL SOUND(10,1000,4)
```

```
1720 CALL RITARDO(1000):: CALL C
LEAR :: CALL CHARSET
1730 SUBEND
1740 SUB RITARDO(RIT)
1750 FOR I=1 TO RIT :: NEXT I
1760 SUBEND
1770 SUB LINO(R,C,RP)
1780 CALL CHAR(129,"FF")
1790 CALL HCHAR(R,C,129,RP):: CA
LL COLOR(13,7,1)
1800 SUBEND
1810 SUB LINV(R,C,RP)
1820 CALL CHAR(128,"404040404040
4040"):: CALL COLOR(13,7,1)
1830 CALL VCHAR(R,C,128,RP)
1840 SUBEND
1850 SUB TITF
1860 DISPLAY AT(2,2):"IN CASA F.
CASA"
1870 DISPLAY AT(2,20):"1"&"  "&"
2"&"  "&"3"
1880 DISPLAY AT(19,2):"1=schedin
a secondo logica"
1890 DISPLAY AT(20,2):"2=schedin
a casuale"
1900 DISPLAY AT(21,2):"3=schedin
a mista"
1910 DISPLAY AT(23,2):"premi un
tasto per finire"
1920 SUBEND
1930 SUB NUM
1940 CL=2 :: UN=1
1950 FOR RG=4 TO 12 STEP 2 :: DI
SPLAY AT(RG,2)SIZE(2):UN :: UN=U
N+2 :: NEXT RG
1960 FOR RG=14 TO 16 STEP 2 :: D
ISPLAY AT(RG,1)SIZE(3):UN :: UN=
UN+2 :: NEXT RG
1970 SUBEND
```

10-90 - Prima pagina (intestazione).
100 - Richiamo programma (titolo).
120-140 - Inizializzazione.
160-260 - Domande preliminari.
320-450 - Loop per le squadre di casa (A, B, C1).
480–570 - Loop per le squadre di casa (C2).
590-710 - Loop per le squadre in trasferta (A, B, C1).
730-800 - Loop per le squadre in trasferta (C2).
850-980 - Ultima pagina con i risultati.
1000-1210 - Elaborazione dati con verifica max di doppie.
1220-1340 - Parte random.
1360-1470 - Parte mista (logica + random).
1490-1730 - Subroutine titolo principale.
1740-1760 - Subroutine ritardo.
1770-1840 - Subroutine linee orizzontali e verticali.
1850-1920 - Subroutine intestazioni ultima pagina.
1930-1970 - Subroutine numeri progressivi della II e III pagina.

Stringhe e variabili

A$ (26,1) - Stringa che contiene i nomi delle squadre.
RI$ (13,1) - Stringa che contiene i risultati da stampare.
B$ (13,1) - Stringa che contiene i risultati random da stampare.
RIS$ (Q,1) - Stringa che contiene i risultati.
MED$ (I,1) - Stringa con i risultati della media da stampare.
RI2 - Somma dati della C2.
RI, RI1 - Somma dati per (A, B, C1)
SC1, SC2 - Contengono i dati da confrontare.
TT, TOT - Totale delle squadre che giocano in (A, B, C1)
GIO, GIO1 - Partite giocate.
X, A, K, AA, C, CC, Q - Contatori.
V, P, S, D, V1, P1, S1, VV, DD - Contengono i dati delle squadre.

■

I Ching: il Libro dei Mutamenti

Interrogate il vostro Texas TI99/4A ed ascoltate i suoi consigli

di Daniela Cerù

Un po' di storia

L'I Ching o Libro dei Mutamenti è un sistema di oracoli che ha origini antichissime: come il nome fa intuire esso è nato in Cina. La tradizione attribuisce la base dell'I Ching ad un saggio, tal Pao-Hsi, vissuto la bellezza di 53 secoli fa. Durante la dinastia degli Yin, circa nell'undicesimo secolo avanti Cristo, il principe Wen e suo figlio Tan inserirono dei loro commenti nel Libro dei Mutamenti. Lo stesso Confucio, in seguito, studiò ed apprezzò l'I Ching.
Ma vediamo che cosa ha a che fare tutto ciò con le pagine di una rivista d'informatica. L'I Ching, come detto, è una sorta d'oracolo che risponde alle domande di chi lo consulta: esso è formato dall'unione di due simboli, "- -" e

Listato 1 - Il programma I Ching.

```
5 DATA 3,4,9,10,13,14,16,17,19,2
0,23,24,26,27
10 DATA 13,14,19,20
15 DIM IC$(64)
20 CALL CLEAR
25 RANDOMIZE
30 C=INT(10*RND)+5
40 CALL COLOR(2,C,C)
50 FOR I=16 TO 17
60 CALL VCHAR(3,I,42,8)
70 NEXT I
80 FOR I=1 TO 14
85 READ R
90 CALL VCHAR(13,R,42,8)
100 NEXT I
110 FOR I=1 TO 4
115 READ R
120 CALL HCHAR(R,5,42,3)
125 CALL HCHAR(R,28,42,3)
130 NEXT I
135 CALL HCHAR(16,11,42,2)
140 CALL HCHAR(17,11,42,2)
145 CALL HCHAR(18,29,42,2)
150 CALL VCHAR(14,21,42,2)
155 CALL VCHAR(15,22,42,2)
160 INPUT " VUOI QUALCHE SPIEGAZ
IONE ? ":R$
170 IF R$="S" THEN 1000
175 IF R$="SI" THEN 1000
180 INPUT "COME TI CHIAMI ?":R$
185 CALL SCREEN(6)
187 CALL COLOR(2,2,6)
190 CALL CLEAR
200 PRINT "BENE ";R$
205 PRINT "Esponimi il tuo probl
ema"
210 PRINT
215 INPUT " ":P$
220 IF LEN(P$)<30 THEN 900
230 IF P$<>"" THEN 215
240 PRINT
245 PRINT "MMMH...HO CAPITO"
250 PRINT "ORA ATTENDI UN ATTIMO
 E..."
254 PRINT
255 PRINT
256 PRINT
270 FOR I=0 TO 5
275 RANDOMIZE
280 A(I)=INT(2*RND)
290 IF A(I)=0 THEN 300 ELSE 320

300 PRINT TAB(12);"** **"
310 GOTO 330
320 PRINT TAB(12);"*****"
330 PRINT
340 FOR J=I*10 TO I*10+9
350 READ IC$(J)
370 NEXT J
380 NEXT I
```

"–" combinati sei volte tra di loro. In tutto ci saranno 64 diverse combinazioni ($2^6 = 64$) ed una di queste sarà la risposta alla domanda formulata.

La sapienza cinese unita alla tecnologia del duemila

Il metodo di consultazione degli antichi cinesi era piuttosto complesso (una scaglia di tartaruga, parecchi steli d'achillea e così via) perciò noi abbiamo pensato di semplificarlo usando un computer.
Già, sarà proprio una di queste fredde macchine, aggregati di diodi e resistenze, di integrati e condensatori, di fili elettrici e di tasti, a dare la risposta ai quesiti che verranno formulati. L'home computer adottato è il TI99/4A della Texas Instruments ma ogni altro personal va bene, sia per la poca memoria di cui c'è bisogno, sia per la facile traduzione che può essere fatta dal BASIC di questa macchina. Come potete vedere dal listato 1 il programma non è affatto lungo; esso è scritto in BASIC normale (non esteso) e le istruzioni grafiche sono quanto mai ridotte. Crediamo però che il bello di questo "pozzo di saggezza" stia soprattutto nelle risposte che vengono date, soprattutto se viene fatta una cosultazione collettiva, tra amici.

Il programma

Diamo, dunque, una rapida occhiata al listato: le istruzioni dalla 5 alla 155 servono per scrivere il titolo del programma e dimensionare il vettore IC$ in cui saranno memorizzati i responsi dell'oracolo. Le righe 25-40 scelgono casualmente un colore tra quelli con codice 5 e 14.
Quindi può essere fornita una breve spiegazione sul Libro dei Mutamenti e sul suo uso per coloro che s'avvicinano per la prima volta a questa fonte di sapienza orientale. Dopo di ciò viene chiesto il nome e terminata questa operazione l'oracolo si mette a vostra disposizione chiedendovi il motivo per cui lo interpellate (180-230); l'esposizione del vostro problema può essere lunga quanto vi pare.
Terminata la vostra richiesta, c'è un attimo di raccoglimento e dopo di ciò l'I Ching darà il suo responso. La pausa è stata ottenuta di proposito (270-380) allo scopo di visualizzare un simbolo alla volta e nel contempo approfittare per leggere una decina dei Data delle righe 1400-1555 che contengono tutti i 64 re-

```
390 FOR I=60 TO 63
400 READ IC$(I)
410 NEXT I
420 PRINT
430 PRINT "ECCO IL MIO RESPONSO
      "
435 N=0
440 FOR I=0 TO 5
450 N=N+A(I)*2^I
455 CALL SOUND(99,110+I*11,1)
460 NEXT I
465 FOR I=0 TO 30
470 CALL SOUND(99,110+I*11,1)
475 NEXT I
480 PRINT IC$(N)
485 I=440
490 CALL SOUND(150,I,1)
495 I=I-10
500 IF I=120 THEN 485
505 CALL KEY(0,X,Z)
510 IF Z=0 THEN 490
515 PRINT TAB(60)
520 PRINT "BENE !Spero che quant
o dettoti sia di aiuto.
   Arrivederci a presto
"
530 END
900 PRINT
910 INPUT "BASTA COSI' ? ":P$
920 IF P$="N" THEN 215
925 IF P$="NO" THEN 215
930 GOTO 240
1000 CALL CLEAR
1005 CALL SCREEN(C)
1007 CALL COLOR(2,2,C)
1010 PRINT "L'I CHING e'un antic
o testo  "
1015 PRINT "cinese,una sorta di
oracolo  "
1020 PRINT "che risponde alle do
mande di "
1025 PRINT "chi lo consulta.
      "
1030 PRINT "Non bisogna pero'chi
edergli  "
1035 PRINT "previsioni metereolo
giche o  "
1040 PRINT "come vincere al toto
calcio o "
1045 PRINT "cose simili.Esso puo
'essere  "
1050 PRINT "utile nel fare una s
celta o  "
1055 PRINT "nel risolvere un'ind
ecisione "
1060 PRINT "I responsi sono sibi
llini ma "
1065 PRINT "degni di un vero ora
colo !!"
1070 CALL KEY(0,X,Z)
1075 IF Z=0 THEN 1070
1100 GOTO 180
```

Seguito listato I Ching.

```
1400 DATA "RICETTIVITA'.SUCCESSO
 CON L'OBBEDIENZA","SGRETOLAMENT
O.CERCA DI NON  MUOVERTI"
1405 DATA "UNIONE.SEI VERAMENTE
SICURO DI TE STESSO ?","CONTEMPL
AZIONE.LO SCOPO NON E'RAGGIUNTO"

1410 DATA "ARMONIA.IL RISPETTO G
ENERA  IL RISPETTO","PROGRESSO.E
SIBITE LE VOSTRE QUALITA'"
1415 DATA "ASSEMBRAMENTO.IL PICC
OLO E' AIUTATO DAL GRANDE","STAS
I.MANCA LA COMUNICAZIONE"
1420 DATA "MODESTIA.BRILLARE MA
SENZA  ABBAGLIARE","LA MONTAGNA.
CONCENTRAZIONE ERIPOSO"
1425 DATA "ARRESTO RAPIDO.COLTIV
ARE UN TALENTO","CRESCITA.OCCORR
E FERMEZZA E CORREZIONE"

1430 DATA "PICCOLI ECCESSI.SUCCE
SSO,MA NELLE PICCOLE COSE","IL V
IAGGIATORE.RICERCA DI UNACCORDO"

1435 DATA "INFLUENZA.SAGACITA'IN
 MOLTE OPINIONI","RITIRATA.LA DI
GNITA'PRIMA DITUTTO"
1440 DATA "COORDINAZIONE.SAGGEZZ
A DELLAESPERIENZA","INESPERIENZA
.INUTILE FARE,SIDEVE RIPETERE"
1445 DATA "L'ABISSO.LA SINCERITA
'E'DA  PREFERIRSI","DISPERSIONE.
COORDINAZIONE  DELLE IDEE"
1450 DATA "VIA LIBERA.TORNATE AI
 MODI  ANTICHI","VERSO LA FINE.I
 GUAI SONO  FINITI"
1455 DATA "OPPRESSIONE.PERCHE TA
NTE PA-ROLE ?","CONFLITTO.BISOGN
A SAPERSI  FERMARE IN TEMPO"
1460 DATA "ASCESA.IL FURBO COGLI
```


TROVERETE NEL PROSSIMO NUMERO

IN EDICOLA DAL 26 GIUGNO

DISEGNATORE: UN PROGRAMMA PER MSX

EASYVIDEO PER C 64

AGENDA TELEFONICA CON IL CASIO

HI-RES PER TI99

ROUTINE 2 PER SPECTRUM

SIMULAZIONE DINAMICA DI FENOMENI CONTINUATIVI CON IL C 64

CUBO MAGICO CON SPECTRUM

STATISTICA A PIÙ DIMENSIONI PER APPLE

DATA BASE PER C 16

Seguito listato I Ching.

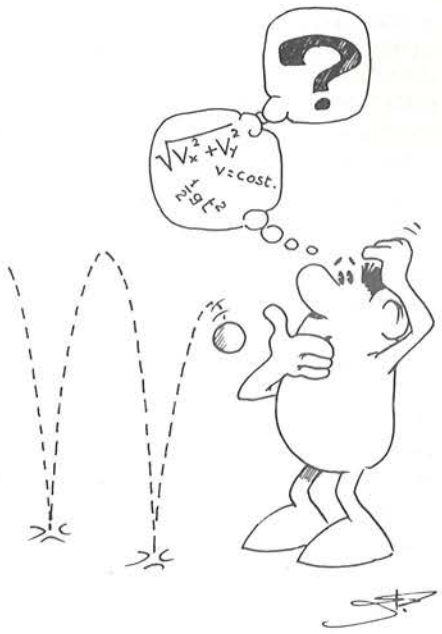
```
E L'OC-CASIONE","PURIFICAZIONE.E
SAMINATE BENELE CONSEGUENZE"
1465 DATA "IL POZZO.AIUTO RECIPR
OCO E  COMPRENSIONE","INSERIMENT
O.BISOGNA TENTARE RIPETUTAMENTE"

1470 DATA "PERSEVERANZA.TUTTE LE
 STRADESONO APERTE","FERMEZZA.TE
NERE GLI OCCHI E LE ORECCHIE BEN
 APERTI"
1475 DATA "GRANDI ECCESSI.INDIFF
ERENZA AI COMMENTI","INCONTRO.NO
N FIDATEVI DEI  PIU'FORTI"
1480 DATA "RITORNO.L'OSTACOLO NO
N E'POICOSI'IMPORTANTE","ALIMENT
O.IL PENSIERO ACCRE- SCE L'AZION
E"
1485 DATA "INIZIO DIFFICILE.TROP
PA SPE-RANZA INGANNA","ACCRESCIM
ENTO.AVANZATE SENZAESITARE"
1490 DATA "IL TUONO.NON METTETEV
I IN  APPRENSIONE","LA PENETRAZ
IONE.IL SAGGIO E'PROTETTO DALLA
LEGGE"
1495 DATA "DISCIPLINA.UN GRANDE
PROTET-TO DAI PICCOLI","INNOCENZ
A.LA SCORRETTEZZA  PORTA AL DIS
ASTRO"
1500 DATA "OSCURAMENTO.PERCHE'AT
TIRARE L'ATTENZIONE ?","AVVENENZ
A.A VOLTE CI SI DEVEFIDARE DELLE
 APPARENZE"
1505 DATA "SUCCESSO.FINIRE E'LA
PARTE  ARDUA","LA FAMIGLIA.OGNI
COSA AL SUOPOSTO"
1510 DATA "ABBONDANZA.IL MEZZO S
UCCESSONON BASTA","CHIAREZZA.BIS
OGNA FARE IL  NECESSARIO"
1515 DATA "RIVOLUZIONE.PRIMA AGI
RE,DOPOPENSARE","COMPAGNIA.NON M
ESCOLATE CIO'CHE NON SI MESCOLA"

1520 DATA "AVVICINAMENTO.IL SUCC
ESSO  PUO'ESSERE PROVVISORIO","
MINORAZIONE.SI STABILIRA'UN CERT
O EQUILIBRIO"
1525 DATA "RESTRIZIONE.FRENO ALL
ENTATO PRESTO O TARDI","SINCERIT
A'.IL SAGGIO CREDE ACOLUI CHE CR
EDE"
1530 DATA "LA FIDANZATA (O LA MO
GLIE). VIRTU'DELL'ASTINENZA","SE
PARAZIONE.LA PREVISIONE  CATTIV
A S'INGANNA"
1535 DATA "GIOIA.NON DIMENTICARE
 ANCHE IL PIACERE","PRUDENZA.MA
FORSE LA TIGRE  NON MORDE"
1540 DATA "PACE.FINE DELLE PICCO
LEZZE EINIZIO DELLE GRANDEZZE","
DOMARE.ISPIRARSI AGLI ALTRI"
1545 DATA "ATTESA.NON FARSI TRAS
CINARE","RINSAVIMENTO.FORSE LA N
UBE  NON SCOPPIERA'"
1550 DATA "POTENZA.LA FERMEZZA T
RIONFA","POSSESSO.TENETE BENE I
FILI IN MANO"
1555 DATA "APERTURA.SPARPAGLIATE
 E NON AMMASSATE","IL CREATIVO.U
N CAMBIAMENTO  E'SALUTARE"
```

sponsi.
A volte può non apparire molto chiaro quello che dice l'I Ching, ma dobbiamo sempre rammentare che ci troviamo di fronte ad un oracolo, il cui scopo è anche indurre coloro che lo consultano ad esaminare sé stessi e le cose, partendo magari dalla riflessione sui suoi responsi (volutamente stringati e sibillini).
La riga 450 calcola in che posizione del vettore IC$ si trova la risposta da dare; quindi verrà prodotta una serie di suoni che durerà finchè l'utente non digita un qualsiasi carattere che decreterà la fine del programma. Come potete vedere è tutto molto semplice e lineare e le uniche istruzioni che i "non texani" dovranno modificare sono le varie Call che agiscono sullo schermo, sul sonoro o sul carattere ricevuto o meno.

Come dettovi in precedenza, noi consigliamo di consultare l'I Ching tra amici perchè possono verificarsi responsi molto interessanti come quanto è successo ad una coppia fresca fresca di matrimonio, di nostra conoscenza, ritornati dal loro viaggio di nozze. A "lei", che chiedeva informazioni sulla fedeltà del marito l'I Ching ha risposto con "Progresso. Esibite le vostre qualità", mentre per "lui" che domandava come sarebbero stati i primi mesi di matrimonio il responso è stato "La fidanzata (o la moglie). Virtù dell'astinenza". Che altro dire?! ■

di Sergio Borsani

Palla che rimbalza



Un giorno, a casa di un amico, ho potuto vedere alcuni giochi di provenienza USA girare su un TI 99/4A equipaggiato con armamento pesante, cioè espansione, floppy, editor/Assembler, speech synthetizer, ecc. In quell'occasione rimasi colpito da un programma che simulava il gioco del tennis. Una voce da Odissea nello Spazio scandiva i punteggi, "fifteen - thirty", "fifteen - forty", e la pallina rimbalzava in modo molto realistico. Potere degli sprite. Sono stati i protagonisti anche all'ultimo corso di BASIC da me organizzato. Non c'è niente che attragga l'attenzione e stimoli la fantasia di un ragazzo, più di uno spiritello che aleggia sullo schermo. Qualcuno è tentato di creare un gioco lì per lì, come se si trattasse di oggetti reali da manipolare con la tastiera invece che con le mani; ma le difficoltà non tardano a venire ed è bene non farsi troppe illusioni.

Il computer o il video che è il mezzo elettivo con il quale ci comunica, ci presentano un microcosmo bidimensionale dalle leggi fisiche alterate; non c'è gravità nè attrito, i corpi, quando si urtano, invece di rimbalzare elasticamente si compenetrano senza deformarsi proseguendo nel loro moto inerziale. Ma che strano moto inerziale. Invece di allontanarsi indefinitamente dal punto di partenza, una volta varcato il bordo dello schermo, ricompaiono esattamente dal lato opposto. Districarsi tra i Print, gli Input e gli If-Then Else, poteva essere un'impresa ardua ma non impossibile, ma qui, per tutti i diavoli, per far saltare una pallina bisogna conoscere la geometria e la fisica, e ricreare a suon di formule il mondo che ci è familiare.

Il libro della Natura è scritto in lingua matematica, diceva Galileo Galei, che non a caso viene qui citato per essere stato il precursore nell'enunciazione delle leggi che regolano la caduta dei gravi e che descrivono le traiettorie dei corpi che si muovono sotto l'azione delle forza di gravità.

"Proiectum, dum fertur motu composito ex horizontali aequabili et ex naturaliter accelerato deorsum, ecc." (Discorsi, IV).

Gli sprite, nel TI 99/4A, hanno la caratteristica, non comune a tutti gli altri computer, di mantenere la loro velocità automaticamente grazie ai parametri presenti nell'istruzione Call Sprite e nella Call Motion. Ad esempio, nel Commodore 64 il movimento avviene modificando continuamente i registri che controllano la posizione.

Una pallina che rimbalza si può immaginare animata contemporaneamente da due movimenti, uno uniforme lungo la direzione orizzontale ed uno uniformemente accelerato lungo la direzione verticale. Nel primo la velocità è costante, la qual cosa si ottiene facilmente nel TI 99/4A definendo una volta per tutte l'intensità della componente orizzontale della velocità. Nel listato 1 è stato posto VX=20, un valore che sposta lo sprite verso destra consentendo qualche rimbalzo prima che la pallina attraversi tutto lo schermo. La componente verticale deve variare secondo la legge Vy=Voy−gt; se inizialmente la pallina viene lanciata verso l'alto, la sua velocità diminuisce di una quantità proporzionale al tempo. Quando, aumentando t, il prodotto gt diventa uguale a Voy, la velocità Vy risulta uguale a zero, la pallina pertanto smette di salire avendo raggiunto il punto più alto della sua traiettoria e poi comincia a scendere. Il nostro computer non possiede la funzione Time ed il passare del tempo dev'essere scandito incrementando una variabile: T=T+1. In fisica se t è espresso in

secondi e V in m/s, la costante g assume il valore 9,7805 0, come riportano la maggior parte dei testi, 9,8. Noi, poiché aumentando T nel modo descritto in precedenza, non abbiamo un tempo espresso in secondi, non potremo usare questo valore di g. Nel programma 1, alla linea 150, abbiamo posto g=2. Perché abbiamo scritto VY+2★T invece di VY−2★T come suggerisce un confronto con la formula data in precedenza?

La ragione sta nel sistema di riferimento adottato. Normalmente l'asse verticale è orientato verso l'alto mentre il computer (non tutti) pone l'origine nell'angolo in alto a sinistra del video e pertanto orienta l'asse verticale verso il basso. A questo si rimedia attribuendo a VY valori negativi e a g valori positivi. Basta un meno o un più per farci andare a gambe all'aria.

Eseguendo il programma vedrete la pallina rimbalzare da sinistra verso destra seguita dalla fedele ombra più in basso. A proposito, come si fa in modo che la pallina non scenda al di sotto della sua ombra? Se controlliamo la posizione dovremo farlo quando la pallina è ancora al di sopra dell'ombra, ma in tal caso il computer non distingue se lo sprite sta scendendo o se sta salendo. Conviene controllare la velocità. Se è negativa la pallina sta ancora salendo, se è positiva, allora sta scendendo. E quando sappiamo che la pallina è prossima al rimbalzo? Quando la componente verticale della velocità è vicina al valore iniziale VY con il segno cambiato; la pallina cioè cade con la stessa velocità con la quale era salita. Poiché inizialmente era VY=−50, se V diventa maggiore di 45 significa proprio che siamo quasi arrivati, poniamo lo sprite al centro dell'ombra con una provvidenziale Call Locate, invertiamo il segno a V, azzeriamo l'orologio e via per un altro rimbalzo.

Qualcuno osserverà giustamente che la pallina prima o dopo dovrà fermarsi; gli urti non sono perfettamente elastici, parte dell'energia viene assorbita e a ogni rimbalzo viene raggiunta una minore altezza mentre la traiettoria si fa meno ampia. La modifica del programma risulta abbastanza agevole, basta inserire un ciclo For-Next che faccia diminuire ad ogni rimbalzo la componente verticale della velocità. Nel programma Palla 2, VY viene fatta variare da −60 a −5 ottenendo così l'effetto desiderato senza ulteriori sostanziali cambiamenti (listato 2).

Che dire poi della palla da tennis dalla quale ha preso inizio il nostro discorso? Una pallina da tennis rimbalza continuamente da sinistra a destra e da destra a sinistra. Le modifiche rispetto al primo programma sono ancora più semplici ed invece di tante spiegazioni proponiamo una specie di quiz. Se qualcuno non trova una soluzione

Listato 1 - Il programma Palla 1.

```
100 REM  PALLA 1
110 CALL CLEAR :: CALL CHAR(128,"071F3F7
F7F"&RPT$("F",12)&"7F7F3F1F07E0F8FCFEFE"
&RPT$("F",12)&"FEFEFCF8E0")
120 CALL CHAR(132,RPT$("0",16)&"0F3F7FFF
7F3F0F"&RPT$("0",18)&"F0FCFEFFFEFCF000")
130 VX=20 :: VY=-50 :: T=1
140 CALL SPRITE(#1,128,12,150,10,VY,VX,#
2,132,2,158,10,0,VX):: CALL MAGNIFY(3)
150 T=T+1 :: V=VY+2*T :: IF V<45 THEN CA
LL MOTION(#1,V,VX):: GOTO 150
160 V=-50 :: T=1 :: CALL POSITION(#2,Y,X
):: CALL LOCATE(#1,150,X):: GOTO 150
```

Listato 2 - Il programma Palla 2.

```
100 REM  PALLA 2
110 CALL CLEAR :: CALL CHAR(128,"071F3F7
F7F"&RPT$("F",12)&"7F7F3F1F07E0F8FCFEFE"
&RPT$("F",12)&"FEFEFCF8E0")
120 CALL CHAR(132,RPT$("0",16)&"0F3F7FFF
7F3F0F"&RPT$("0",18)&"F0FCFEFFFEFCF000")
130 VX=10 :: VY=-60 :: T=1
140 CALL SPRITE(#1,128,12,150,10,VY,VX,#
2,132,2,158,10,0,VX):: CALL MAGNIFY(3)
150 FOR VY=-60 TO -5 STEP 5
160 T=T+1 :: V=VY+2*T :: IF V<ABS(VY+4)T
HEN CALL MOTION(#1,V,VX):: GOTO 160
170 V=VY :: T=1 :: CALL POSITION(#2,Y,X)
:: CALL LOCATE(#1,150,X)
180 NEXT VY :: T=1 :: GOTO 140
```

Listato 3 - Il programma Palla 3.

```
100 REM  PALLA 3
110 CALL CLEAR :: CALL CHAR(128,"071F3F7
F7F"&RPT$("F",12)&"7F7F3F1F07E0F8FCFEFE"
&RPT$("F",12)&"FEFEFCF8E0")
120 CALL CHAR(132,RPT$("0",16)&"0F3F7FFF
7F3F0F"&RPT$("0",18)&"F0FCFEFFFEFCF000")
130 VX=30 :: VY=-50 :: T=1
140 CALL SPRITE(#1,128,12,150,10,VY,VX,#
2,132,2,158,10,0,VX):: CALL MAGNIFY(3)
150 T=T+1 :: V=VY+2*T :: IF V<45 THEN 17
0
160 V=-50 :: T=1 :: CALL POSITION(#2,Y,X
):: CALL LOCATE(#1,150,X):: VX=-VX :: CA
LL MOTION(#2,0,VX)
170 CALL MOTION(#1,V,VX):: GOTO 150
```

Listato 4 - Il programma Palla 4.

```
100 REM  PALLA 4
110 CALL CLEAR :: CALL CHAR(128,"071F3F7
F7F"&RPT$("F",12)&"7F7F3F1F07E0F8FCFEFE"
&RPT$("F",12)&"FEFEFCF8E0")
120 CALL CHAR(132,RPT$("0",16)&"0F3F7FFF
7F3F0F"&RPT$("0",18)&"F0FCFEFFFEFCF000")
130 VX=60 :: VY=-50 :: T=1
140 CALL SPRITE(#1,128,12,150,10,VY,VX,#
2,132,2,158,10,0,VX):: CALL MAGNIFY(3)
150 T=T+1 :: V=VY+4*T :: IF V<45 THEN 17
0
160 V=-50 :: T=1 :: CALL POSITION(#2,Y,X
):: CALL LOCATE(#1,150,X):: VX=-VX :: CA
LL MOTION(#2,0,VX)
170 CALL MOTION(#1,V,VX):: GOTO 150
```

soddisfacente può esaminare il listato 3, ma è invitato a farlo non prima di aver eseguito qualche tentativo.

Vogliamo infine aggiustare il tiro. Dopo tutto la risposta contenuta nel programma 3 sembra piuttosto un "pallonetto". Sapreste ottenere un colpo teso e all'apparenza dotato di una maggiore energia? Un consiglio: aumentare VX e simulare un campo gravitazionale più intenso. Non arrendetevi subito, una possibile soluzione, è contenuta nel listato 4. ■

di Paolo Agostini

File Examiner

Molto spesso succede di incontrare difficoltà nell'esaminare programmi in BASIC o in linguaggio macchina, a causa del fatto che hanno qualche sistema di protezione oppure perché nel corso del caricamento, vengono posizionati in locazioni di memoria difficilmente accessibili (Per esempio RAM posta sotto la memoria ROM), o altro.

Per ovviare a tale inconveniente, abbiamo progettato il programma Filex per il C 64 che ha le seguenti funzioni:

1) legge la directory del disco senza disturbare la memoria del computer;

2) legge da disco e visualizza i programmi in BASIC convertendone i "token" in parole-chiave;

3) legge da disco e disassembla i programmi in linguaggio macchina, visualizzandoli sullo schermo sotto forma di codice in linguaggio Assembly;

4) legge da disco e visualizza file numerici o di altro tipo, quali ad esempio i file di un word processor.

Il programma, proprio a causa della diversificazione dei compiti che si propone, è abbastanza lungo, e necessita della massima attenzione nella trascrizione dei dati relativi ai codici operativi. Noterete infatti la presenza, in coda ad ogni codice, di quella che viene chiamata dagli addetti ai lavori "opcode tag", cioè di una lettera che segue le tre lettere del codice operativo e che serve a specificarne il "modo":

1) codice di 3 lettere seguito da uno spazio

Listato 1 - Il programma Filex.

```
5 POKE53280,4:POKE53281,7:PRINTCHR$(147);
  CHR$(31);"ATTENDERE PREGO"
7 FORX=1TO18:X$=X$+CHR$(32):NEXTX:GOSUB10
  00
9 REM ------------------------------
10 REM ***  MENU PRINCIPALE ***
11 REM ------------------------------
20 PRINTCHR$(147);TAB(7);CHR$(176);:FORI=
   1TO23:PRINTCHR$(192);:NEXT
30 PRINTCHR$(174)
40 PRINTTAB(7);CHR$(221);SPC(5);"FILE EXA
   MINER";SPC(5);CHR$(221)
50 PRINTTAB(7);CHR$(221);SPC(5);"DI P.AGO
   STINI";SPC(5);CHR$(221)
60 PRINTTAB(7);CHR$(173);:FORI=1TO23:PRIN
   TCHR$(192);:NEXT
70 PRINTCHR$(189)
80 PRINT:PRINT:PRINTTAB(17);CHR$(18);"MEN
   U";CHR$(146)
90 PRINT:PRINT:PRINT
100 PRINTTAB(6);CHR$(18);" (1) ";CHR$(146
    );SPC(2);"LEGGI LA DIRECTORY"
110 PRINT:PRINTTAB(6);CHR$(18);" (2) ";CH
    R$(146);SPC(2);"LEGGI IL FILE"
130 PRINT:PRINTTAB(6);CHR$(18);" (3) ";CH
    R$(146);SPC(2);"FINE"
140 PRINT:PRINT:PRINTTAB(16);CHR$(18);"QU
    ALE?";CHR$(146)
150 GETA$:A=VAL(A$):IFA<1ORA>3THEN150
160 ONAGOTO 200,500,3200
170 END
179 REM ------------------------------
180 REM CHIUSURA CANALI APERTI
181 REM ------------------------------
190 PRINT#15,"I0":CLOSE1:CLOSE15:GOTO3100
199 REM ------------------------------
200 REM **** LEGGE LA DIRECTORY ****
201 REM ------------------------------
210 PRINTCHR$(147):PRINT"DISK NAME = ";
220 CLOSE1:CLOSE15:FL=0:L$="":B$=""
230 OPEN1,8,0,"$":OPEN15,8,15
240 INPUT#15,E1,E1$,E2,E3:IF E1 THENPRINT
     E1,E1$:GOTO190
250 FORI=1TO33:GET#1,A$
260 L$=L$+A$:NEXT:CLOSE1:PRINTL$
270 FORI=1TO40:PRINTCHR$(45);:NEXTI
280 PRINT"{RVS}BLOCKS{ 3 SPAZI}FILE NAME
    { 6 SPAZI}TIPO "
290 OPEN1,8,0,"$"
300 GET#1,A$:GET#1,A$:GOSUB350
310 GOSUB350:IF FL=1 THENPRINT:GOTO190
320 PRINTRIGHT$(X$+STR$(L),4);CHR$(32);CH
    R$(32);CHR$(32);
330 PRINTLEFT$(B$+X$,18);FT$:GETA1$:IFA1$
    <>""THEN3070
340 GOTO310
349 REM ------------------------------
350 REM SUBROUTINE LETTURA DIRECTORY
351 REM ------------------------------
360 B$="":GET#1,A$:GET#1,A$
370 GET#1,A$:L=ASC(A$+CHR$(0))
380 GET#1,A$:L=L+ASC(A$+CHR$(0))*256
390 GET#1,A$:INPUT#15,E1,E1$,E2,E3:IF E1
    THENPRINT E1,E1$:GOTO190
400 IFA$=""THENFL=1:RETURN
410 IFA$<>CHR$(34)THEN390
420 GET#1,A$
425 IFPEEK(653)THEN425
430 IFA$=CHR$(34)THEN450
440 B$=B$+A$:GOTO420
450 GET#1,A$:IFA$=CHR$(32)THEN450
460 FT$=A$
```

Hi-Res

Un programma per disegnare e stampare in alta risoluzione

di Sergio Borsani

Nel n. 14 di **Personal Software** - Gennaio 1984 - apparve un articolo intitolato *Grafici ad alta risoluzione con il TI99/4A*. Seguirono molte telefonate di lettori che non riuscivano a far "girare" il programma che, nonostante tutto, funzionava benissimo. Speriamo che tutti, prima o poi, siano riusciti ad individuare gli errori di trascrizione. I controlli, a volte, sono veramente estenuanti.

Supponiamo che ormai sia nota a tutti la tecnica per ottenere l'alta risoluzione. I caratteri del TI99/4A sono facilmente ridefinibili con l'istruzione Char; così, almeno in teoria, si può controllare il video pixel per pixel. Diciamo in teoria perché il set di caratteri, da solo, non è sufficiente a coprire l'intero schermo. Secondo il tipo di BASIC usato, si dispone di 111 o 127 caratteri, escludendo quello di codice ASCII 32 che è lo space, mentre il video possiede 32 x 24 = 768 locazioni. È un peccato che nel TI BASIC manchino le istruzioni per gestire la grafica in bitmap, tanto più se si considera che i nuovi computer dello standard MSX usano lo stesso microprocessore della Texas Instruments per il controllo del video e possiedono tutte le più comuni istruzioni grafiche, come Pset, Line, Circle, Paint, ecc.

Nel precedente programma di Gennaio, solo apparentemente il grafico di una funzione occupava l'intero video; in realtà la maggior parte della superficie era costituita da Space. Nel caso di un grafico più "denso", utilizzati i caratteri disponibili, l'elaborazione si sarebbe in-

Listato 1 - Il programma Hi-Res: ogni figura presenta un listato che indica il sottoprogramma utente da inserire nel blocco principale.

```
100 REM ******************
110 REM *                  *
120 REM *      HI-RES      *
130 REM *                  *
140 REM ******************
150 REM *                  *
160 REM * COPYRIGHT 1984 *
170 REM *                  *
180 REM *  BY S.BORSANI    *
190 REM *                  *
200 REM ******************
210 REM *                  *
220 REM *     TI-99/4A     *
230 REM *                  *
240 REM *     VERSIONE     *
250 REM *                  *
260 REM * EXTENDED BASIC *
270 REM *                  *
280 REM ******************
290 GOTO 390
300 CALL CLEAR :: CALL CHAR :: CALL HCHA
R :: CALL VCHAR :: DIM S$(15,4),E$(15,4)
,M$(100)
310 DATA 8,4,2,1,9,5,3,1,A,6,2,3,B,7,3,3
,C,4,6,5,D,5,7,5,E,6,6,7,F,7,7,7
320 DATA 8,C,A,9,9,D,B,9,A,E,A,B,B,F,B,B
,C,C,E,D,D,D,F,D,E,E,E,F,F,F,F,F
330 DATA 0,0,0,0,1,1,1,0,2,2,0,2,3,3,1,2
,4,0,4,4,5,1,5,4,6,2,4,6,7,3,5,6
340 DATA 0,8,8,8,1,9,9,8,2,A,8,A,3,B,9,A
,4,8,C,C,5,9,D,C,6,A,C,E,7,B,D,E
350 J,K,FINE,N,CTR,MODO,X,Y,X1,Y1,S,XK,Y
K,Q,CO,RO,NC,LN,BI,PO,POX,A,NT,T,NUMCAR=
LOOP
360 NAME$,RECORD$,SC$,W$,A1$,DN$,FN$=" "
 :: CALL CHARSET :: CALL COLOR :: CALL K
EY :: CALL GCHAR :: CALL CHARPAT
370 CALL LOCATE :: CALL SPRITE :: CALL D
ELSPRITE
380 !@P-
390 FOR J=0 TO 15 :: FOR K=1 TO 4 :: REA
D S$(J,K):: NEXT K :: NEXT J
400 FOR J=0 TO 15 :: FOR K=1 TO 4 :: REA
D E$(J,K):: NEXT K :: NEXT J :: GOTO 440
410 CALL CHAR(33,"00000000000000FF"):: C
ALL CHAR(34,"0101010101010101")
420 CALL CHAR(35,"8080808080808080"):: C
ALL CHAR(36,"FF"):: CALL CHAR(37,"3C4299
A1A199423C")
430 CALL CHAR(38,"FF818181818181FF"):: C
ALL CHAR(39,"806070381C0E0702"):: RETURN
440 CALL CHAR(140,"0042241818244200"):: 
CALL CHAR(141,"007C42427C4442")
450 CALL CHAR(142,"00183C7E181818"):: CA
LL CHAR(143,"001818187E3C1800")
460 FOR J=1 TO 100 :: M$(J)=RPT$("O",16)
:: NEXT J :: FINE=17
470 CALL CLEAR :: CALL CHARSET :: GOSUB
410 :: FOR J=2 TO 14 :: CALL COLOR(J,2,1
):: NEXT J
480 DISPLAY AT(2,7):"H  I  -  R  E  S" :
: DISPLAY AT(4,2):"COPYRIGHT ";CHR$(37);
"1984 S. BORSANI"
490 CALL HCHAR(1,3,36,28):: CALL VCHAR(1
,2,34,5):: CALL VCHAR(1,31,35,5):: CALL
HCHAR(5,3,33,28)
```

terrotta. Un modo per ovviare a questo inconveniente consiste nel definire una finestra grafica tale da essere completamente occupata da caratteri ridefinibili, potendo così disporre a piacimento di tutti i pixel in essa contenuti.

Il programma presentato utilizza, appunto, una finestra grafica di 10 x 10 caratteri ovvero di 80 x 80 pixel. In essa si creano immagini o con l'aiuto della tastiera o con un sottoprogramma scritto dall'utente.

Si possono anche memorizzare le immagini in un file residente su nastro o su disco, richiamare in memoria vecchi disegni per eventuali modifiche ed infine stampare il disegno su carta. Il programma, insomma, si comporta come un semplice editor di immagini. La configurazione minima, richiede solo il BASIC esteso; la stampante non è indispensabile, in sua assenza ci si può accontentare dell'output sul video.

Come si utilizza il programma

Il menu consiste nelle seguenti opzioni: Load, Draw, Save, Print, Erase e Files. Come suggeriscono i termini stessi, la prima carica in memoria un file nel quale è memorizzato un disegno precedentemente registrato con l'opzione Save; la seconda fa apparire la finestra grafica sulla quale viene realizzato il disegno; la terza registra un file contenente i pattern dei caratteri che formano il disegno; la quarta produce una copia su carta; la quinta azzera la matrice adoperata per memorizzare il disegno; la sesta, infine, funzionante solo con un sistema a dischi, mostra i nomi dei file presenti sul dischetto.

Per eseguire un disegno, si seleziona l'opzione 2 del menu. Appare la finestra grafica, un cursore a forma di matita e quattro simboli nella parte bassa del video. Il loro significato, nell'ordine, è: Pen Down (la penna scrive), Pen Up (la penna si sposta senza scrivere), Erase (la penna cancella), Return (si torna al menu). Il simbolo attivo è racchiuso da una cornice rossa ed inizialmente tale sorte tocca a Pen Up. Se esiste un sottoprogramma scritto dall'utente, questo viene eseguito subito, prima di poter disegnare con la tastiera. Si parlerà più avanti di questa possibilità. Per iniziare il disegno bisogna conoscere alcuni comandi fondamentali. Il cursore grafico a forma di matita si può trovare all'interno o all'esterno della finestra; il movimento nei due sensi avviene alla pressione del tasto Enter. Quando il cursore si trova all'esterno della finestra i tasti con le frecce <S> e <D> attivano una funzione operativa spostando la cornice sul simbolo corrispondente, quando invece il cursore si trova all'interno, otto tasti provvedono al suo movimento nelle direzioni principali e lungo le diagonali.

```
500 DISPLAY AT(8,4):"1 LOAD" :: DISPLAY
AT(10,4):"2 DRAW" :: DISPLAY AT(12,4):"3
 SAVE"
510 DISPLAY AT(14,4):"4 PRINT" :: DISPLA
Y AT(16,4):"5 ERASE" :: DISPLAY AT(18,4)
:"6 FILES"
520 DISPLAY AT(20,10):"SCELTA?" :: ACCEP
T AT(20,18)SIZE(1)VALIDATE("123456"):SC$
530 ON ERROR 1350 :: N=VAL(SC$):: CALL H
CHAR(6+2*N,4,38):: ON N GOTO 540,580,107
0,1110,1170,1200
540 GOSUB 1420 :: IF NAME$="CS1" THEN CA
LL CLEAR
550 OPEN #1:NAME$,SEQUENTIAL,INPUT ,DISP
LAY ,FIXED 80
560 FOR J=1 TO 20 :: INPUT #1:RECORD$ ::
 FOR K=1 TO 5 :: M$((J-1)*5+K)=SEG$(RECO
RD$,(K-1)*16+1,16):: NEXT K :: NEXT J
570 CLOSE #1 :: IF NAME$="CS1" THEN 470
ELSE GOSUB 1410 :: GOTO 520
580 CALL CLEAR :: GOSUB 1490 :: FOR J=2
TO 14 :: CALL COLOR(J,2,16):: NEXT J ::
CTR=-1
590 FOR J=8 TO 17 :: FOR K=12 TO 21 :: C
TR=CTR+1 :: CALL HCHAR(J,K,40+CTR):: NEX
T K :: NEXT J :: CALL UTENTE
600 MODO=2 :: X=127 :: Y=97 :: X1=113 ::
 Y1=169 :: GOSUB 1360 :: GOTO 860
610 CALL KEY(0,K,S):: IF S=0 THEN 610
620 IF K=13 THEN 860
630 IF K=69 THEN XK=0 :: YK=1 :: GOTO 71
0
640 IF K=82 THEN XK,YK=1 :: GOTO 710
650 IF K=68 THEN XK=1 :: YK=0 :: GOTO 71
0
660 IF K=67 THEN XK=1 :: YK=-1 :: GOTO 7
10
670 IF K=88 THEN XK=0 :: YK=-1 :: GOTO 7
10
680 IF K=90 THEN XK,YK=-1 :: GOTO 710
690 IF K=83 THEN XK=-1 :: YK=0 :: GOTO 7
10
700 IF K=87 THEN XK=-1 :: YK=1 ELSE 610
710 Q=Y-YK :: IF Q>56 AND Q<137 THEN Y=Q
720 Q=X+XK :: IF Q>88 AND Q<167 THEN X=Q
730 IF MODO=2 THEN 840
740 CO=INT((X-1)/8)+1 :: RO=INT((Y-1)/8)
+1
750 CALL GCHAR(RO,CO,NC):: CALL CHARPAT(
NC,A1$):: LN=Y-INT((RO-1)*8):: BI=X-INT(
(X-1)/4)*4 :: PO=INT((X-1)/4)+1
760 IF PO/2=INT(PO/2)THEN POX=LN*2 ELSE
POX=LN*2-1
770 W$=SEG$(A1$,POX,1):: IF W$="F" AND M
ODO=1 THEN 840
780 IF W$="0" AND MODO=3 THEN 840
790 A=ASC(W$):: IF A>60 THEN NT=A-55 ELS
E NT=A-48
800 IF MODO=1 THEN W$=S$(NT,BI)ELSE W$=E
$(NT,BI)
810 IF POX=1 THEN A1$=W$&SEG$(A1$,2,15):
: GOTO 840
820 IF POX=16 THEN A1$=SEG$(A1$,1,15)&W$
 :: GOTO 840
830 A1$=SEG$(A1$,1,POX-1)&W$&SEG$(A1$,PO
X+1,16-POX)
840 CALL LOCATE(#1,Y,X):: IF MODO=2 THEN
 610
850 CALL CHAR(NC,A1$):: GOTO 610
860 CALL LOCATE(#1,145,140):: FOR T=1 TO
```

Come è ovvio, in queste applicazioni i tasti sono quelli con le frecce ed i loro vicini: E = alto, X = basso, S = sinistra, D = destra, R = alto-destra, C = basso-destra, Z = basso-sinistra, W = alto-sinistra. È stato previsto l'uso esclusivo della tastiera poiché i joystick, che teoricamente sembravano possedere le carte migliori, all'atto pratico si sono rivelati poco sensibili agli spostamenti in diagonale.

Un disegno, generalmente richiede un frequente cambio di funzioni, ma con un po' di pratica non è difficile controllare il movimento del cursore grafico, farlo uscire dalla finestra con il tasto Enter, cambiare la funzione e tornare nello stesso punto con il cursore per riprendere il disegno.

Quando il lavoro è terminato, per tornare al menu, bisogna attivare il simbolo R sulla destra e attendere brevemente perché il computer memorizzi le stringhe esadecimali che definiscono i caratteri, prima che essi assumano nuovamente il loro aspetto abituale. Dopo questa fase appare una linea alla base della finestra; la linea può essere spostata con i tasti Fctn E e Fctn X rispettivamente verso l'alto e verso il basso. Lo scopo è quello di "tagliare" la finestra riducendo in tal modo la superficie da stampare. Chi invece desidera stampare tutta la finestra o non intende affatto eseguire la stampa, premerà semplicemente il tasto Enter, tornando così al menu principale. La precedente operazione per limitare la superficie da stampare si rende necessaria per la relativa lentezza con la quale viene eseguita la copia su carta. Se il disegno non occupa tutta la finestra e si è avuta l'accortezza di realizzarlo nella parte alta, un "taglio" farà risparmiare in seguito del tempo prezioso impedendo la stampa di inutili caratteri vuoti.

Il sottoprogramma utente

Una parte interessante del programma è certamente quella rivolta al sottoprogramma che l'utente può scrivere e inserire nel listato, dopo la linea 2100, non dimenticando di terminare con l'istruzione Subend. Per facilitare la programmazione sono state create tre nuove istruzioni grafiche con altrettanti sottoprogrammi BASIC. Esse sono: Pset per segnare un punto; Line per tracciare una linea e Circle per tracciare cerchi ed ellissi. Queste nuove istruzioni, come tutti i sottoprogrammi, vengono chiamate con delle Call rispettando una precisa sintassi.

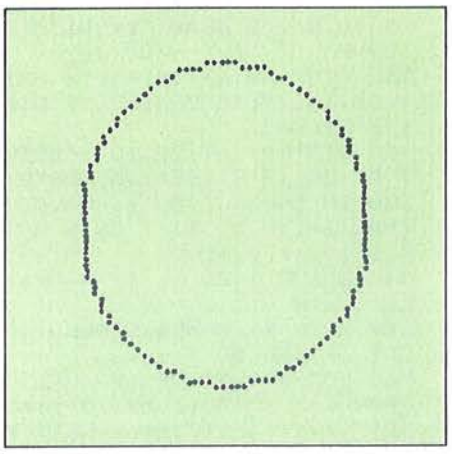
```
2050 REM ************************
2060 REM *                      *
2070 REM * SOTTOPROGRAMMA UTENTE *
2080 REM *          GLOBO       *
2090 REM ************************
2100 SUB UTENTE
2110 CALL CIRCLE(39,39,30,1):: CALL LINE
(39,9,39,69):: CALL CIRCLE(39,39,30,.3)
2120 FOR J=1.2 TO .3 STEP -.3 :: R=INT(S
IN(J)*30):: E=30/R :: CALL CIRCLE(39,39,
(R),(E)):: NEXT J
2130 SUBEND
```

Figura 1 - Immagine di un globo creata con un sottoprogramma che utilizza le nuove istruzioni Line e Circle. Le lettere N e S sono state aggiunte controllando con la tastiera il cursore grafico.

Call Pset (x,y) attiva un pixel della finestra grafica colorandolo di nero. L'origine del sistema di riferimento è posto nell'angolo in basso a sinistra ed i punti sono numerati dallo 0 al 79. I valori esterni a tale intervallo non provocano alcun errore, ma vengono semplicemente ignorati.

Call Line (x1, y1, x2, y2) traccia una linea dal punto di coordinate x1, y1 al punto di coordinate x2, y2. Se entrambi i punti cadono all'esterno della finestra l'istruzione viene ignorata, se invece uno solo degli estremi è esterno, allora viene tracciato solo il segmento che rientra nella finestra grafica.

Figura 2 - A causa della diversa distanza dei punti che formano l'immagine nelle due direzioni, verticale ed orizzontale, una circonferenza, calcolata matematicamente, appare come un'ellisse. Una diversa deformazione si ha in fase di stampa per la diversa distanza degli aghi della testina della stampante.
La figura mostra la riproduzione di un cerchio che appariva tale sul video, ottenuto con un'eccentricità uguale a 1,38.



Call Circle (x0, y0, r, e) traccia un cerchio di centro x0, y0, di raggio r e di eccentricità e. Anche qui esiste la funzione di "clipping" per tracciare ugualmente la curva qualora una parte dovesse cadere all'esterno della finestra. Per quanto riguarda l'eccentricità, da un punto di vista geometrico, il valore 1 corrisponde ad una circonferenza, valori diversi da 1 ad ellissi. In realtà i punti che formano il video non sono equidistanti nelle due direzioni, orizzontale e verticale, con il risultato che un'eccentricità uguale a 1 fa apparire un'ellisse. Un

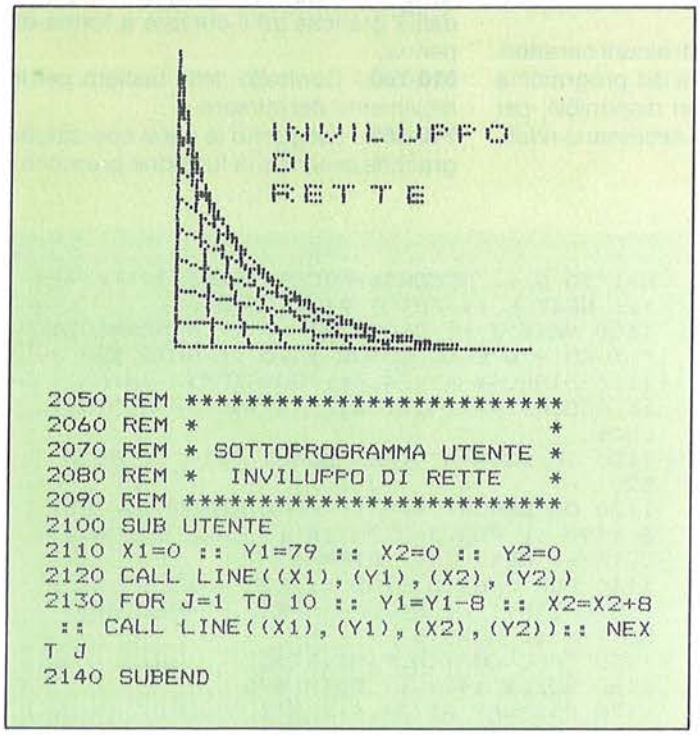
valore di 1,39 dà buoni risultati sullo schermo, facendo apparire un cerchio; purtroppo in fase di stampa, a causa della diversa conformazione della matrice dei punti, si ottiene nuovamente un'ellisse, anche se non molto eccentrica. In ogni caso, mancando un valore di default, bisogna indicare il valore che si ritiene più opportuno.

La figura 2 mostra appunto un cerchio che appariva veramente tale sul video; durante la stampa ha subito una leggera deformazione.

Le figure 3 e 4 mostrano immagini ottenute con altrettanti sottoprogrammi usando istruzioni Line e Circle, completate poi tramite la tastiera.

La stampa

Per quanto riguarda la stampa, è necessario conoscere alcune caratteristiche della stampante. Al momento dell'acquisto la testina di stampa è regolata dalla fabbrica in modo da lavorare con 7 aghi di stampa. Tuttavia, per poter sfruttare pienamente le sue capacità grafiche è necessario tarare in posizione Off l'interruttore dip SW2-1 posto sul quadro dell'interfaccia seriale. L'operazione è semplice, e per eseguirla basta seguire le istruzioni contenute nel manuale. Se non sono state apportate altre modifiche alla taratura degli interruttori dip della stampante, tutto funzionerà regolarmente. Spesso, tuttavia, viene cambiata la velocità di trasmissione dati che la Casa ha fissato a 300 baud. Con velocità diverse l'utente deve modificare la linea 1880 del programma, contenente l'istruzione Open #4: "RS232.CR.DA = 8", per adattarla alle proprie necessità.




```
2050 REM ************************
2060 REM *                      *
2070 REM * SOTTOPROGRAMMA UTENTE *
2080 REM *  INVILUPPO DI RETTE   *
2090 REM ************************
2100 SUB UTENTE
2110 X1=0 :: Y1=79 :: X2=0 :: Y2=0
2120 CALL LINE((X1),(Y1),(X2),(Y2))
2130 FOR J=1 TO 10 :: Y1=Y1-8 :: X2=X2+8
 :: CALL LINE((X1),(Y1),(X2),(Y2)):: NEX
T J
2140 SUBEND
```

Figura 3 - Inviluppo di rette ottenuto con un ciclo For-Next nel quale viene chiamato il sottoprogramma Sub Line. Il testo è stato inserito controllando il cursore grafico con la tastiera. Infatti, creando una finestra in bit map non si dispone più del normale set di caratteri alfanumerici.

Come si ottiene l'indirizzamento indipendente dei punti?

Prima di terminare, merita una breve spiegazione il modo in cui viene messo in On un pixel della finestra grafica.

I punti del video non sono indirizzabili indipendentemente poiché la memoria è organizzata in modo da contenere, da un lato, i numeri dei caratteri presenti in ciascuna locazione dello schermo, dall'altro la forma dei caratteri stessi, codificata, a livello utente, con stringhe di 16 caratteri esadecimali. In pratica, per mettere in On un pixel, bisogna ridefinire il carattere al quale appartiene. Se osserviamo le linee del programma a partire dalla numero 740, per prima cosa vengono trovati i numeri di colonna e di riga (CO e RO) della locazione dove si trova il cursore. Il numero di codice ASCII del carattere (NC) viene ricavato con l'istruzione Call Gchar mentre la successiva Call Chrpat resistuisce il pattern del carattere, cioè la stringa di caratteri esadecimali (A1$) ognuno dei quali controlla un semibyte in memoria,

```
2050 REM ************************
2060 REM *                      *
2070 REM * SOTTOPROGRAMMA UTENTE *
2080 REM *     FUNZIONE  SIN    *
2090 REM ************************
2100 SUB UTENTE
2110 CALL LINE(0,39,79,39):: CALL LINE(3
9,0,39,79)
2120 Y1=INT(SIN(-39/4)*20+39):: FOR X=0
TO 79 :: Y2=INT(SIN((X-39)/4)*20+39)
2130 CALL LINE((X),(Y1),(X),(Y2)):: Y1=Y
2 :: NEXT X
2140 SUBEND
```

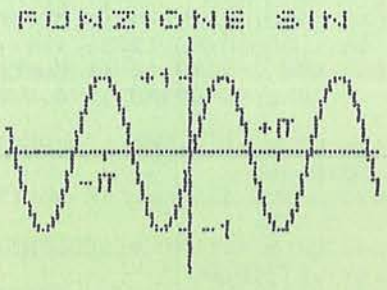


Figura 4 - Esempio di grafico di funzione.

ovvero una sequenza di 4 pixel. Individuato il semibyte, si tratta di modificare un bit lasciando inalterati gli altri tre. A questo punto si presenta un'alternativa: passare dalla notazione esadecimale a quella decimale, usare gli operatori booleani e poi tornare nuovamente all'esadecimale, oppure creare delle tabelle di composizione (una sorta di tavole pitagoriche) per operare direttamente con caratteri esadecimali. Nel programma è stata adottata questa seconda soluzione. Le tabelle sono contenute nelle matrici S$() e E$() e servono rispettivamente per mettere in On o in Off un bit e conseguentemente un pixel del video. Individuta la posizione del carattere esadecimale nella stringa (POX), il suo valore decimale corrisponde ad una riga della tabella, mentre il numero del bit (BI) all'interno del semibyte, corrisponde ad una colonna della tabella. L'elemento S$(NT,BI) restituisce il nuovo carattere esadecimale che codifica il semibyte modificato. Ricostruita infine la stringa che costituisce il pattern del carattere, apparirà un nuovo punto sullo schermo.

Descrizione del listato

300-380 - Elenco delle variabili per il prescan. È questa una tecnica per abbreviare il tempo d'attesa all'inizio del programma.
390-400 - Caricamento delle tabelle per effettuare operazioni logiche, in esadecimale, tra bit e semibyte corrispondenti. In pratica tali tabelle serviranno per porre in On o in Off un pixel della finestra grafica.
410-460 - Definizione di alcuni caratteri. Poiché la parte grafica del programma assorbe tutti i caratteri disponibili, per poter riporre il menu, è necessario ridefinire alcuni caratteri.
470-530 - Menu. La linea 470 ristabilisce il normale set di caratteri alfanumerici.
540-570 - Eseguono il comando Load. La Gosub di linea 540 chiede il nome del file.
580-590 - In esecuzione del comando Draw, appare la finestra grafica e vengono ridefiniti i caratteri (subroutine 1490) per far apparire il disegno attualmente in memoria e conservato nel vettore M$(). Prima di proseguire viene lanciato il sottoprogramma Sub Utente. Se questo non contiene istruzioni si passa all'esecuzione del disegno sotto il controllo della tastiera.
600 - Chiama una subroutine per far apparire i simboli per selezionare le modalità grafiche ed il cursore a forma di penna.
610-700 - Controllo della tastiera per il movimento del cursore.
710-850 - Eseguono le varie operazioni grafiche secondo la funzione prescelta.

Seguito programma Hi-Res

```
 50 :: NEXT T
870 CALL KEY(0,K,S):: IF S=0 THEN 870
880 IF K=13 THEN 950
890 IF K=68 AND MODO<4 THEN MODO=MODO+1
:: GOTO 910
900 IF K=83 AND MODO>1 THEN MODO=MODO-1
:: GOTO 910 ELSE 870
910 IF MODO=1 THEN X1=97 :: GOTO 940
920 IF MODO=2 THEN X1=113 :: GOTO 940
930 IF MODO=3 THEN X1=129 ELSE X1=153
940 CALL LOCATE(#2,Y1,X1):: GOTO 860
950 IF MODO<>4 THEN 730
960 FOR J=1 TO 100 :: CALL CHARPAT(J+39,
A1$):: M$(J)=A1$ :: NEXT J
970 CALL DELSPRITE(ALL)
980 CALL HCHAR(18,12,36,10):: RO=18
990 CALL KEY(0,K,S):: IF S=0 THEN 990
1000 IF K=13 THEN FINE=RO-1 :: GOTO 470
1010 IF K=11 AND RO>9 THEN 1040
1020 IF K=10 AND RO<18 THEN 1050
1030 GOTO 990
1040 CALL HCHAR(RO,12,32,10):: RO=RO-1 :
: CALL HCHAR(RO,12,36,10):: GOTO 990
1050 FOR J=1 TO 10 :: NUMCAR=(RO-8)*10+3
9 :: CALL HCHAR(RO,11+J,NUMCAR+J):: NEXT
 J
1060 RO=RO+1 :: CALL HCHAR(RO,12,36,10):
: GOTO 990
1070 GOSUB 1420 :: IF NAME$="CS1" THEN C
ALL CLEAR
1080 OPEN #1:NAME$,SEQUENTIAL,OUTPUT,DIS
PLAY ,FIXED 80
1090 FOR J=1 TO 20 :: RECORD$="" :: FOR
K=1 TO 5 :: RECORD$=RECORD$&M$((J-1)*5+K
):: NEXT K :: PRINT #1:RECORD$
1100 NEXT J :: CLOSE #1 :: IF NAME$="CS1
" THEN 470 ELSE GOSUB 1410 :: GOTO 520
1110 DISPLAY AT(24,6):"CONFERMI? (Y/N)"
:: ACCEPT AT(24,22)SIZE(1)VALIDATE("YN")
:SC$
1120 IF SC$="N" THEN GOSUB 1410 :: GOTO
520
1130 ON ERROR 1440 :: CALL CLEAR :: GOSU
B 1490 :: FOR J=2 TO 14 :: CALL COLOR(J,
2,16):: NEXT J :: CTR=-1
1140 FOR J=8 TO FINE :: FOR K=12 TO 21 :
: CTR=CTR+1 :: CALL HCHAR(J,K,40+CTR)::
NEXT K :: NEXT J
1150 CALL COPY(8,FINE,12,21)
1160 GOSUB 1410 :: GOTO 470
1170 DISPLAY AT(24,6):"SEI SICURO? (Y/N)
" :: ACCEPT AT(24,24)SIZE(1)VALIDATE("YN
"):SC$
1180 IF SC$="N" THEN GOSUB 1410 :: GOTO
520
1190 FOR J=1 TO 100 :: M$(J)=RPT$("0",16
):: NEXT J :: GOSUB 1410 :: GOTO 520
1200 ON ERROR 1470 :: CALL CLEAR :: OPEN
 #3:"DSK1.",INPUT ,RELATIVE,INTERNAL ::
CTR=0
1210 INPUT #3:DN$,J,J,K :: PRINT "DISKNA
ME=";DN$:"AVAILABLE=";K;"USED";J-K
1220 FOR LOOP=1 TO 127 :: INPUT #3:FN$,Y
,J,K :: IF LEN(FN$)=0 THEN 1280
1230 IF ABS(Y)=2 THEN PRINT FN$,:: CTR=C
TR+1
1240 IF CTR<40 THEN 1270
1250 PRINT :"PRESS <ENTER> TO CONTINUE"
```

In particolare, le linee 740-760 individuano il carattere nel quale si trova il cursore ed il semibyte nel quale si trova il bit da modificare.
860-940 - Controllo della tastiera per la selezione di una funzione grafica (Pen Up, Pen Down, ecc.).
950-960 - Il disegno viene memorizzato nel vettore M$.
970-1060 - Si permette il "taglio" della finestra grafica in modo da impedire la stampa di Space con un'inutile perdita di tempo.
1070-1100 - Esecuzione del comando Save con il quale viene memorizzato su nastro o su disco il vettore M$ contenente i pattern dei caratteri che, in precedenza, formavano il disegno.
1110-1160 - La sezione effettua l'hard copy su carta dell'immagine presente nella finestra grafica. A tal fine viene chiamato il sottoprogramma Sub Copy. Nella Call, tra parentesi, vengono trasmessi i numeri di riga e di colonna che delimitano la finestra grafica.
1170-1190 - Azzeramento degli elementi del vettore M$ e conseguentemente del disegno presente in memoria.
1200-1340 - Lettura della directory per la lista dei file presenti sul dischetto ed eventuale cancellazione di un file.

Subroutine

1350 - Subroutine legata ad una On Error. Rimanda il programma al menu.
1360-1400 - Fa apparire i simboli per selezionare le modalità grafiche ed il cursore a forma di penna.
1410 - Cancella la parte inferiore del video.
1420-1430 - Accetta il nome del file contenente un'immagine da memorizzare.
1440-1460 - Subroutine per la gestione degli errori.
1470-1480 - Messaggio conseguente all'uso del comando Files con il registratore a cassette. Tale opzione è consentita solo con il sistema a dischi.
1490-1500 - Ridefinisce i caratteri con i pattern memorizzati nel vettore M$.

Sottoprogrammi

1530-1670 - Sub Pset. Pone in On un pixel del video dopo aver individuato il carattere ed il semibyte corrispondenti.
1680-1780 - Sub Line. Traccia una linea utilizzando il precedente sottoprogramma.
1790-1860 - Sub Circle. Traccia un cerchio.
1870-2040 - Sub Copy. Esegue una copia su carta del disegno presente nella finestra grafica. La linea 1880 contiene la Open per l'interfaccia seriale. Si presuppone che la taratura degli interruttori dip della stampante sia quella originale ad eccezione di quello che attiva parole di 8 bit (vedi articolo).
2050-2110 - Sub Utente. Nel listato que-

Seguito programma Hi-Res

```
1260 CALL KEY(0,K,S):: IF S=0 THEN 1260
ELSE PRINT :: CTR=0
1270 NEXT LOOP
1280 CLOSE #3 :: PRINT :"DETETE A FILE?
(Y/N) N" :: ACCEPT AT(23,22)SIZE(-1)VALI
DATE("YN"):SC$
1290 IF SC$="N" THEN 470
1300 PRINT :: PRINT :: PRINT
1310 GOSUB 1420 :: DISPLAY AT(24,6):"SEI
 SICURO? (Y,N) N"
1320 ACCEPT AT(24,24)SIZE(-1)VALIDATE("Y
N"):SC$ :: IF SC$="N" THEN 470
1330 OPEN #2:NAME$,SEQUENTIAL,INPUT ,DIS
PLAY ,FIXED 20 :: CLOSE #2:DELETE :: GOT
O 470
1340 END
1350 CALL DELSPRITE(ALL):: RETURN 470
1360 CALL HCHAR(22,13,143):: CALL HCHAR(
22,15,142):: CALL HCHAR(22,17,140):: CAL
L HCHAR(22,20,141)
1370 CALL HCHAR(21,12,36,10):: CALL HCHA
R(23,12,33,10)
1380 CALL VCHAR(21,11,34,3):: CALL VCHAR
(21,22,35,3)
1390 CALL SPRITE(#1,39,7,145,140,0,0,#2,
38,9,Y1,X1,0,0)
1400 RETURN
1410 CALL HCHAR(23,3,32,60):: CALL VCHAR
(8,4,32,12):: RETURN
1420 ON ERROR 1440 :: DISPLAY AT(23,1):"
FILE NAME?" :: ACCEPT AT(23,12)SIZE(15):
NAME$
1430 RETURN
```

```
1440 DISPLAY AT(24,1):"* ERROR * PRESS <
ENTER>"
1450 CALL KEY(0,K,S):: IF S=0 THEN 1450
1460 IF K=13 THEN GOSUB 1410 :: RETURN 5
20 ELSE 1450
1470 PRINT "ONLY FOR DISK SYSTEM!":" ":"
PRESS <ENTER> TO CONTINUE"
1480 CALL KEY(0,K,S):: IF S=0 OR K<>13 T
HEN 1480 ELSE RETURN 470
1490 FOR J=40 TO 139 :: CALL CHAR(J,M$(J
-39)):: NEXT J
1500 RETURN
1510 !@P+
1520 END
1530 SUB PSET(X,Y)
1540 X=INT(X+.5):: Y=INT(Y+.5)
1550 IF X<0 OR X>79 THEN SUBEXIT
1560 IF Y<0 OR Y>79 THEN SUBEXIT
1570 RO=17-INT(Y/8):: CO=INT(X/8)+12 ::
CALL GCHAR(RO,CO,NC):: CALL CHARPAT(NC,P
$)
1580 LN=(INT(Y/8)+1)*8-Y :: BI=X-INT(X/4
)*4 :: PO=INT(X/4)+1
1590 IF PO/2=INT(PO/2)THEN POX=LN*2 ELSE
 POX=LN*2-1
1600 W$=SEG$(P$,POX,1):: IF W$="F" THEN
SUBEXIT
1610 A=ASC(W$):: IF A>60 THEN DEC=A-55 E
LSE DEC=A-48
1620 DEC=DEC OR 2^(3-BI):: IF DEC<10 THE
N W$=STR$(DEC)ELSE W$=CHR$(DEC+55)
1630 IF POX=1 THEN P$=W$&SEG$(P$,2,15)::
 GOTO 1660
1640 IF POX=16 THEN P$=SEG$(P$,1,15)&W$
:: GOTO 1660
```

sto sottoprogramma non contiene alcuna istruzione; per il buon funzionamento, la sua presenza è ugualmente necessaria.

Principali variabili del programma

S$() - Tabella a due dimensioni per mettere in On un bit di un semibyte espresso nella notazione esadecimale.
E$() - Tabella simile alla precedente usata per mettere in Off un bit.
M$() - Vettore usato per memorizzare i pattern dei caratteri quando questi devono essere resettati.
Name$ - Nome del dispositivo usato come memoria di massa (nastro o disco).
Record$ - Record del file costituito dal pattern di 5 caratteri. Si ricorda che un pattern viene codificato con una stringa di 16 caratteri esadecimali.
SC$ - Scelta dell'utente al menu.
W$, A1$ - Variabili di lavoro usate per ottenere la stringa di caratteri esadecimali che ridefiniscono un carattere dopo che è stato modificato mettendo in On un pixel.
DN$ - Variabile utilizzata per leggere il nome del dischetto dalla directory.
FN$ - Variabile usata per leggere i nomi dei file dalla directory.
J,K - Variabili di lavoro adoperate prevalentemente nei cicli.
Fine - Contiene il numero di riga del video che delimita la superficie da stampare.
Modo - Assume i valori 1, 2, 3 e 4, corrispondenti ai quattro simboli posti al di sotto della finestra grafica.
X, Y, X1, Y1 - Coordinate di un punto all'interno della finestra grafica.
XK, YK - Assumono i valori –1, 0, + 1 per indicare lo spostamento relativo del cursore grafico.
Q - Variabile di comodo per il calcolo di una nuova coordinata.
CO,RO - Numero di colonna e numero di riga.
NC - Numero di codice ASCII di un carattere.
LN - Una delle otto linee di otto punti che formano un carattere nella quale si trova il cursore grafico.
BI - Bit, di un semibyte, che va posto in On.
PO - Indica la posizione del semibyte specificando se è di sinistra o di destra.
POX - Indica la posizione di un carattere esadecimale all'interno della stringa che definisce un carattere.
NT - Numero di un elemento delle tabelle contenute nelle matrici S$ e E$.
Numcar - Numero di un carattere presente nella finestra grafica. ■

Seguito programma HI-Res

```
1650 P$=SEG$(P$,1,POX-1)&W$&SEG$(P$,POX+
1,16-POX)
1660 CALL CHAR(NC,P$)
1670 SUBEND
1680 SUB LINE(X1,Y1,X2,Y2)
1690 IF X1=X2 THEN X=X1 :: GOTO 1760
1700 IF X1>X2 THEN Q=X1 :: X1=X2 :: X2=Q
 :: Q=Y1 :: Y1=Y2 :: Y2=Q
1710 YO=Y1 :: CALL PSET((X1),(Y1))
1720 FOR X=X1 TO X2
1730 Y=INT((X-X1)*(Y2-Y1)/(X2-X1))+Y1 ::
 W=INT(Y-YO):: IF ABS(W)>1 THEN Y=YO+SGN
(W):: FLAG=1
1740 CALL PSET((X),(Y)):: YO=Y :: IF FLA
G=1 THEN FLAG=0 :: GOTO 1730
1750 NEXT X :: SUBEXIT
1760 IF Y1>Y2 THEN Q=Y1 :: Y1=Y2 :: Y2=Q
1770 FOR Y=Y1 TO Y2 :: CALL PSET(X,Y)::
NEXT Y
1780 SUBEND
1790 SUB CIRCLE(XO,YO,R,E)
1800 Y1=YO+R*E :: CALL PSET((XO),(Y1))
1810 FOR X=XO TO XO+R
1820 Y=YO+SQR(R*R-(XO-X)^2)*E :: W=INT(Y
-Y1):: IF ABS(W)>1 THEN Y=Y1+SGN(W):: FL
AG=1
1830 CALL PSET((X),(Y)):: CALL PSET(X,2*
YO-Y):: CALL PSET((2*XO-X),(Y)):: CALL P
SET((2*XO-X),(2*YO-Y))
1840 Y1=Y :: IF FLAG=1 THEN FLAG=0 :: GO
TO 1820
1850 NEXT X
1860 SUBEND
1870 SUB COPY(R1,R2,C1,C2)
1880 OPEN #4:"RS232.CR.DA=8" :: PRINT #4
:CHR$(27);CHR$(65);CHR$(8);
1890 FOR R=R1 TO R2 :: FOR C=C1 TO C2 ::
 CALL GCHAR(R,C,N):: CALL CHARPAT(N,W$)
1900 PRINT #4:CHR$(27);"K";CHR$(8);CHR$(
0);
1910 FOR K=3 TO 0 STEP -1 :: Q=0 :: FOR
E=1 TO 15 STEP 2
1920 D=ASC(SEG$(W$,E,1)):: IF D<60 THEN
D=D-48 ELSE D=D-55
1930 D=D AND 2^K :: IF D THEN Q=Q+2^(7-I
NT(E/2))
1940 NEXT E :: PRINT #4:CHR$(Q);
1950 NEXT K
1960 FOR K=3 TO 0 STEP -1 :: Q=0 :: FOR
E=2 TO 16 STEP 2
1970 D=ASC(SEG$(W$,E,1)):: IF D<60 THEN
D=D-48 ELSE D=D-55
1980 D=D AND 2^K :: IF D THEN Q=Q+2^(8-I
NT(E/2))
1990 NEXT E :: PRINT #4:CHR$(Q);
2000 NEXT K
2010 NEXT C :: PRINT #4:CHR$(13);CHR$(10
);
2020 NEXT R
2030 PRINT #4:CHR$(27);CHR$(65);CHR$(12)
;:: CLOSE #4
2040 SUBEND
2050 REM ************************
2060 REM *                      *
2070 REM * SOTTOPROGRAMMA UTENTE *
2080 REM *                      *
2090 REM ************************
2100 SUB UTENTE
2110 SUBEND
```

FAI CRESCERE IL TUO TI 99/4A CON LA NUOVA FORMIDABILE OFFERTA J.SOFT

Dopo il successo ottenuto dalla prima proposta, ecco a disposizione del tuo TI-994/A un assortimento di altri splendidi programmi, tra i quali il richiestissimo EXTENDED BASIC; il MODULO ADVENTURE, che ti permetterà di vivere appassionanti avventure e l'EDITOR/ASSEMBLER, indispensabile per il definitivo salto di qualità del tuo computer. Ordina subito, per essere sicuro di ricevere i programmi al più presto.

EXTENDED BASIC
Cod. DTXNX01 - L. 270.000

Finalmente disponibile il linguaggio di programmazione compatibile con il BASIC TI-99/4A, con le seguenti aggiunte: righe multidichiarazioni, dichiarazioni IF-THEN-ELSE, accesso diretto allo schermo, sottoprogrammi BASIC con variabili locali, autoavviamento dei programmi su disco, facile controllo di fino a 28 sprite con concatenamento programmi BASIC, fusione codici, programmi protetti, dichiarazioni ON ERROR (Errore in atto) e funzioni logiche di Boole (AND, OR, NOT 1E, O, NO, ecc.).
Permette, con l'espansione della memoria, di programmare in linguaggio macchina.

MODULO AVVENTURA
Cod. DTXRX01 - L. 70.000

Si tratta del modulo di comando, da inserire nella consolle, che funge da interprete del linguaggio delle cassette sottodescritte.
Uno strumento indispensabile per arricchire la tua software-teca di nuove entusiasmanti avventure.
Il modulo viene fornito con l'avventura "IL PIRATA".
Configurazione richiesta: registratore a cassetta.

Elenco delle "Avventure" che girano esclusivamente col MODULO AVVENTURA

Nove programmi entusiasmanti, semplici da usare ma difficili da finire.
All'inizio di ogni avventura viene proposta la situazione in cui ci si trova e viene posta la domanda: COSA DEVO FARE? Utilizzando gli oltre 200 vocaboli previsti dal vocabolario del computer, darai gli ordini conseguenti, per trovarti in situazioni sempre diverse fino a realizzare lo scopo del gioco, ogni volta differente per ogni titolo.
Avventure che stimolano la tua capacità di ragionare in modo logico e che richiedono ore, e anche settimane, per concludersi.

Mystery Fon House (La casa del Mistero)
Cod. DTXMX17 - L. 30.000

Prima di procedere all'esplorazione, devi meditare su come fare per entrare nella Casa del Mistero.
Una volta dentro, potrai vedere tutti gli angoli della casa, dove si nasconde un prezioso premio.

Pyramid of Doom (La Piramide del Destino)
Cod. DTXMX22 - L. 30.000

L'avventura inizia in un deserto, nei pressi di uno stagno dal quale fuoriesce un palo. Grazie alle tue ricerche, troverai una piramide solo di recente scoperta dalle sabbie mobili.
Cercherai l'ingresso, raccoglierai tesori e cercherai di fuggire dalla piramide.

Strange Odyssey (Strana Odissea)
Cod. DTXMX24 - L. 30.000

Inizia quando ti rendi conto di essere "naufragato" su un minuscolo pianeta e quindi devi riparare l'astronave per poter tornare a casa.
Mentre cerchi i pezzi di ricambio sul pianeta, tenterai di scoprire i segreti di un'antica civiltà, raccogliendo i tesori in cui ti imbatterai lungo la strada.

Adventureland
Cod. DTXMX01 - L. 30.000

Ti trovi all'inizio della foresta di un mondo di fiaba. Esplorandolo, potrai scoprire 13 diversi tesori come pure il luogo particolare dove riporli.

Voodoo Castle (Castello del Voodoo)
Cod. DTXMX32 - L. 30.000

Sei all'interno di una cappella di fronte ad una bara chiusa. Esplorando il castello, cercherai di raccogliere le informazioni necessarie per liberare il Conte Cristo dalla terribile maledizione che i suoi nemici gli hanno fatto.

Ghost Town (La Città Fantasma)
Cod. DTXMX09 - L. 30.000

Tutte le città fantasma sono misteriose, ma questa ha un fascino particolare: è abitata da fantasmi veri ed è piena di tesori.
Esplora tutti i vecchi edifici per vedere quanti ne riuscirai a trovare.

Savage Island Series (Serie dell'Isola Selvaggia)
Cod. DTXMX35 - L. 30.000

L'avventura si divide in due parti ed inizia ai margini di una giungla impenetrabile. Mentre esplori l'isola, incontrerai delle creature insolite. Dopo aver completato la prima parte, riceverai la parola d'ordine per iniziare la seconda parte, dove o diventerai il più grande eroe al mondo o andrai incontro a una fine rapida ed orribile.

The Count (Il conte)
Cod. DTXMX26 - L. 30.000

Ti svegli da un pisolino e ti trovi in un letto con un paletto da tenda in mano. Sta a te scoprire chi sei, cosa stai facendo in Transilvania e perché il postino ha consegnato un flacone di sangue!

The Golden Voyage
Cod. DTXMX10 - L. 30.000

Comincerai con l'individuare il palazzo reale nella città persiana. Nel palazzo incontrerai l'anziano re al quale rimangono solo tre giorni di vita, a meno che non riuscirai a ringiovanirlo.
Portando soltanto una borsa d'oro, salperai in mare alla ricerca della mitica fontana della giovinezza.

I programmi seguenti sono invece moduli Solid State Software, pronti da inserire:

Music Maker (Il compositore di Musica)
Cod. DTXQX01 - L. 42.000 - con istruzioni in italiano

Un ottimo programma sia per chi vuole avvicinarsi alla musica senza cognizioni di base, sia per chi deve studiare composizione direttamente sul programma e sia per chi utilizza il suono dal punto di vista tecnico ed ha interesse per tipo di frequenza, ecc.. Il programma prevede due opzioni: scrittura di partiture sul pentagramma: inserendo battuta per battuta è possibile scrivere partiture composte anche da tre voci e persino stamparne gli spartiti (in questo caso è necessaria una stampante e l'interfaccia RS232 o similare). L'altra opzione prevede la realizzazione delle melodie prendendo in esame gli Hertz di una singola nota.

TI Invaders (Invasori TI)
Cod. DTXMX27 - L. 30.000

Il mondo viene attaccato da numerose creature odiose provenienti dallo spazio. Devi fare affidamento sulla tua agilità mentale e fisica per distruggere le creature multicolori con i missili a disposizione.

Parsec
Cod. DTXMX21 - L. 42.000

PARSEC è la tua astronave, con la quale combatterai gli alieni ribelli e le loro navicelle, buttandole fuori rotta e emetterai un fuoco dal tuo laser che fa inaridire. Tenterai di sopravvivere alla "cintura" di asteroidi mortali.
Per attivare le capacità sonore di questa cartuccia, puoi utilizzare lo Speech Synthesizer.

Soccer (Gioco del Calcio)
Cod. DTXMX23 - L. 37.000

Inutile spendere troppe parole per questo apprezzatissimo programma. Cinque giocatori per squadra, con passaggi, tiri, intercettamenti, parate, cariche e molte altre tattiche del calcio. Potete persino avere il replay immediato di una rete, al rallentatore.

Car Wars (Guerre di Auto)
Cod. DTXMX06 - L. 30.000

Se ti emozionano le corse automobilistiche, abbinate alla possibilità di battere l'avversario con astute manovre, ecco il programma che fa per te. Cimenta la tua velocità ed abilità contro il computer mentre cerchi di avanzare sul circuito, evitando incidenti che farebbe saltare il tuo bolide fuori pista.

E ora un'altra eccezionale proposta:

Game Writer's Pack 1 & 2
Cod. DTXIX13 - L. 35 000
(prezzo per le 2 confezioni)

Un'ottima realizzazione didattica divisa in due parti autonome ma complementari che insegna i principi (e li approfondisce) della programmazione BASIC, tramite la programmazione di videogiochi.

Molto ben strutturata e comprensibile, utilizza un linguaggio semplice ed elementare. In ogni confezione è compresa una cassetta con i giochi di cui si parla nel corso della dissertazione. In totale 6 giochi per confezione.
La trattazione è basata sul TI-BASIC anche se non disdegna l'EXTENDED BASIC.

Fun - Pac (1-2-3)
Cod. DTXMX36 - L. 27.000
(prezzo per 3 cassette)

Tre cassette (non acquistabili separatamente), con 8 giochi complessivi ben strutturati ed avvincenti. Dalla corsa dei cavalli al gioco del golf. Dal salvataggio di scienziati dalla contaminazione nucleare alla navicella Supernova, un ottimo gioco "adventure"

E INFINE...

Editor/Assembler
Cod. DTXAX01 - L. 180.000

L'eccezionale programma che ti permette di programmare in linguaggio di programmazione TSM 9900 è dà accesso diretto a tutte le caratteristiche del sistema tra cui audio, voce, grafica ed I/O (Entrata/Uscita), oltre a mettere a disposizione la massima rapidità grazie al microprocessore a 16 bit del computer. Le routine in linguaggio di programmazione possono essere svolte come programmi indipendenti oppure collegate a programmi TI BASIC o TI BASIC ESTESO con l'impiego di una chiamata di sottoroutine. Oltre al modulo SSS, il programma comprende due floppy-disk (uno dei quali contiene il gioco TOMBSTONE CITY) e il Manuale Utente che dà tutte le informazioni sulla composizione del software.
L'EDITOR/ASSEMBLER richiede i seguenti optional:

— **Peripheral Expansion Box**
Cod. ETX0X01 - L. 320.000

— **32 K Expansion Card**
Cod. ETXOX02 - L. 340.000

- **Drive Control Card**
Cod. ETXOX03 - L. 440.000

— **Floppy disk drive Box**
Cod. ETXOX04 - L. 480.000

N. B. È possibile acquistare insieme gli optional ETXOX01 - ETXOX02 - ETXOX03 - ETXOX04, al prezzo complessivo di L. 1.250.000 anziché L. 1.580.000. In questo caso, il codice da indicare per l'acquisto dei 4 pezzi è ETXOX05

e, infine, freschi di stampa, escontati del 10% sul prezzo di copertina:

— **Paper Book** - TI-99/4A (con cassetta) - editore *J.soft*
Cod. ASOC007 - L. 16.200

Una selezione dei migliori programmi tratti dal nostro settimanale PAPER SOFT, scelti accuratamente sulla base della loro originalità e fantasia.

— **35 Programmi per il TI-99/A4** - editore MUZZIO
Cod. AMUCO24 - L. 9.000

Programmi divertenti e didattici, dal labirinto al tennis, dagli anagrammi alle conversioni metriche e tanti altri ancora.

RITAGLIA ED INVIA IL TAGLIANDO A J.SOFT

Spett. J.*soft* - Viale Restelli, 5 - 20124 Milano (MI)
Tel. 02/6888228-683797-6880841/2/3/

Ordino il seguente software/libri per il TI-99/4A:

cod. ..... cod. ..... cod. .....
cod. ..... cod. ..... cod. .....
cod. ..... cod. ..... cod. .....

per un totale di L. ..... + L. 2.500 per contributo fisso di spese di spedizione.

Scelgo la seguente modalità di pagamento:

- ☐ pagherò in contrassegno al postino
- ☐ allego assegno (o contanti)
- ☐ verso l'importo sul C.C.P. n. 19445204 intestato a *J.soft* (allego ricevuta)

Nome .....

Cognome .....

Via ..... n. .....

CAP ..... Città ..... Prov. .....

di Sergio Borsani

Quali segreti ci riserva la stampa?

In questa puntata parlerò della stampante. So che molti non posseggono questo dispositivo, d'altra parte credo che una buona percentuale di coloro non sappiano sfruttarne le grandi capacità. Chi scrive dispone di una Epson MX-80 Type III, ma non fa alcuna differenza dal momento che il TI99/4A usa lo stesso modello rimarchiato Texas Instruments. Detto per inciso, lo stesso vale per l'IBM PC.

Le caratteristiche di stampa sono ben illustrate nel manuale, per cui cercherò di porre l'accento sui particolari meno evidenti piuttosto che su alcune possibilità alle quali non viene fatto alcun riferimento sullo stesso manuale della Texas, ma che sono presenti in quello originale della Epson.

Il computer comunica con la stampante tramite una porta seriale RS-232; pertanto ogni operazione richiede l'apertura di un canale con una Open. Diverso è il caso in cui si usi un'interfaccia parallela poiché in tal caso la scrittura o il listato di un programma si effettuano semplicemente con i comandi/istruzioni Lprint e Llist. Nel TI99/4A il trasferimento dei dati su stampante avviene previa istruzione Open # 1:"RS-232" o, per il listato, List "RS-232". Alla sigla RS-232, tra virgolette, dovranno seguire alcuni parametri a seconda di come sono predisposti i dip-switch interni della stampante. Se non vengono modificati i dip-switch sarà sufficiente la dicitura "RS-232". Tuttavia è molto frequente una diversa taratura della velocità di trasmissione dei dati e l'attivazione di otto fili di stampa.

La velocità di trasmissione è fissata dalla casa a 300 bps (bit al sec) e sono attivati solo 7 fili di stampa. Una diversa taratura richiede una esplicita dichiarazione nella Open. Ad esempio, Open # 1: "RS-232.BA = 4800.DA = 8" significa che la stampante è stata predisposta per ricevere dati alla velocità di 4.800 bps e che processa parole da otto bit (al posto di sette). Negli esempi e nei listati che seguiranno si userà l'istruzione Open # 1:"RS-232.DA = 8", ma è evidente che ognuno dovrà adattarla alle proprie esigenze. Chi non avesse attivato 8 fili di stampa non incapperà in alcuna situazione d'errore, ma non potrà sfruttare completamente le capacità grafiche della stampante. Buon per lui che qui si parla solo del modo testo.

Parliamo ora dei caratteri di controllo. La stampa avviene in diverse modalità, con caratteri ingranditi, condensati, in grassetto, enfatizzati e ancora con diversa larghezza del modulo oppure in modo testo o in modo grafico a singola e doppia densità, ecc. Tutto si ottiene inviando opportuni caratteri di controllo. Pensiamo che tutti sappiano che Print # 1:Chr$(14); attiva la stampa in doppia larghezza e che Print # 1:Chr$(15); attiva invece la stampa di caratteri condensati, 132 caratteri per riga. A volte si può restare imbarazzati di fronte a certe prestazioni. **Personal Software,** ed altre riviste, richiedono possibilmente listati su 40 colonne, per ovvie esigenze di impaginazione. Come ottenere la stampa su 40 colonne? È semplice, basta inviare i seguenti caratteri di controllo: Print # 1:Chr$(27);Chr$(81);Chr$(40);.

Listato 1 - Prova di stampa con caratteri normali, ingranditi, condensati, enfatizzati e doppi.

```
100 OPEN #1:"RS232.DA=8"
110 PRINT #1:"CARATTERI NORMALI"
120 PRINT #1:CHR$(14);
130 PRINT #1:"CARATTERI INGRANDITI"
140 PRINT #1:CHR$(20);CHR$(15);
150 PRINT #1:"CARATTERI CONDENSATI"
160 PRINT #1:CHR$(18);
162 PRINT #1:CHR$(14);CHR$(15);
164 PRINT #1:"CARATTERI INGRANDITI E CONDENSATI"
166 PRINT #1:CHR$(20);CHR$(18);
170 PRINT #1:CHR$(27);CHR$(69);
180 PRINT #1:"CARATTERI ENFATIZZATI"
190 PRINT #1:CHR$(27);CHR$(70);
200 PRINT #1:CHR$(27);CHR$(71);
210 PRINT #1:"CARATTERI DOPPI"
220 PRINT #1:CHR$(27);CHR$(72);
230 PRINT #1:CHR$(27);CHR$(69);CHR$(14);
240 PRINT #1:"CARATTERI INGRANDITI ED ENFATIZZATI"
250 PRINT #1:CHR$(20);
260 PRINT #1:CHR$(27);CHR$(71);
270 PRINT #1:"CARATTERI ENFATIZZATI E DOPPI"
280 PRINT #1:CHR$(27);CHR$(70);
290 PRINT #1:CHR$(27);CHR$(72);
300 CLOSE #1
310 END
```

```
CARATTERI NORMALI
CARATTERI  INGRANDITI
CARATTERI CONDENSATI
CARATTERI INGRANDITI E CONDENSATI
CARATTERI ENFATIZZATI
CARATTERI DOPPI
CARATTERI  INGRANDITI  ED  ENFATIZZATI
CARATTERI ENFATIZZATI E DOPPI
```

Chr$(81) si riferisce alla larghezza del modulo, Chr$(40) al numero di colonne. In attesa di tali parametri il TI99 invia automaticamente un comando di ritorno carrello (CR) ogni 80 caratteri.
Qualcuno si sarà accorto che, pur specificando una larghezza di stampa di 132 colonne, ottenibile con caratteri condensati, si ha ugualmente un ritorno carrello dopo 80 colonne. La ragione sta in quanto detto in precedenza e per utilizzare tutta la larghezza del modulo bisogna inibire il CR automatico con: Open #1:"RS-232.DA = 8.CR". Tuttavia, annullato il ritorno automatico, sarà necessario provvedere manualmente, per così dire, con i caratteri di controllo Chr$(13); Chr$(10), corrispondenti rispettivamente al ritorno carrello ed all'avanzamento riga (vedi listato 2.)

Listato 2 - Prova di stampa in compresso.

```
100 OPEN #1:"RS232.DA=8.CR"
110 PRINT #1:CHR$(15);CHR$(27);CHR$(81);CHR$(132);
120 FOR J=1 TO 6
130 PRINT #1:"STAMPA SU 132 COLONNE ";
140 NEXT J
150 PRINT #1:CHR$(13);CHR$(10);CHR$(18)
160 CLOSE #1
170 END
```

```
STAMPA SU 132 COLONNE STAMPA SU 132 COLONNE STAMPA SU 132 COLONNE STAMPA SU 132 COLONNE STAMPA SU 132 COLONNE STAMPA SU 132 COLONNE
```

Un'altra caratteristica che riteniamo non venga molto usata, è quella che consente la stampa di dati sotto forma di tabelle. Lo schermo del TI99/4A è ahimé limitato con le sue 28 colonne di testo e volendo tabulare i dati elaborati ci si trova subito nei pasticci. In un programma pubblicato recentemente da **Personal Software**, *Ripartizione spese di riscaldamento*, il prospetto di ripartizione veniva scomposto sul video in varie parti comprendenti due o al massimo tre colonne di dati. Il problema naturalmente non si pone sulla stampante, a patto di saper usare convenientemente i codici di tabulazione orizzontale. La stampante riconosce fino a 12 punti di arresto per la tabulazione ed ignora i successivi se ne vengono indicati in numero maggiore. I codici da inviare sono: Print #1:Chr$(27);Chr$(68);Chr$(n1);Chr$(n2);...;Chr$(0), dove n1, n2, ... sono i punti di arresto espressi in numero di colonna. Per tabulare un dato si antepone ad esso il codice Chr$(9);. Prima di provare il listato 3 assicuratevi di annullare i codici dati in precedenza, relativi alla larghezza del modulo e alla stampa condensata. Il breve programma usa l'istruzione Print Using che si rende indispensabile qualora i dati siano in forma numerica; la corretta sintassi per l'output sulla stampante è: Print #1,Using "# # # # .#":N;. Tra virgolette, dopo la parola chiave Using, c'è il formato di stampa che naturalmente può essere cambiato a seconda delle esigenze.

Ciò che manca assolutamente nel manuale della Texas Instruments è il riferimento alla sottolineatura e ai caratteri sovrascritti (esempio: esponenti nelle formule matematiche) e sottoscritti (esempio: indici). Per la sottolineatura bisogna inserire l'opzione CR nella Open per impedire il ritorno carrello automatico; dopo di che si può retrocedere il carrello, senza eseguire

Listato 3 - Prova di tabulazione orizzontale.

```
100 OPEN #1:"RS232.DA=8"
110 DATA 345,54.8,22,15.3,999,456.3,728.
4,9813
120 DATA 1.4,55.6,234,67.9,578.9,47,123.
6,67
130 DATA 44,12.7,8365,87.3,87.1,599,834.
6,6.2
140 PRINT #1:CHR$(27);CHR$(68);CHR$(10);
CHR$(20);CHR$(30);CHR$(40);CHR$(50);CHR$
(60);CHR$(70);CHR$(0);
150 FOR K=1 TO 3
160 READ N
170 PRINT #1,USING "#####.#":N;
180 FOR J=1 TO 7
190 READ N
200 PRINT #1:CHR$(9);
210 PRINT #1,USING "#####.#":N;
220 NEXT J
230 PRINT #1
240 NEXT K
250 CLOSE #1
260 END
```

```
345.0      54.8      22.0      15.3     999.0     456.3     728.4    9813.0
  1.4      55.6     234.0      67.9     578.9      47.0     123.6      67.0
 44.0      12.7    8365.0      87.3      87.1     599.0     834.6       6.2
```

Listato 4 - Esempio di sottolineatura.

```
100 OPEN #1:"RS232.DA=8.CR"
110 PRINT #1:"TI-99/4A HOME COMPUTER";
120 PRINT #1:CHR$(13);
130 PRINT #1:RPT$("_",22);
140 PRINT #1:CHR$(13);CHR$(10)
150 CLOSE #1
160 END

TI-99/4A HOME COMPUTER
```

l'avanzamento riga, con il codice Chr$(13) e sovrastampare la lineetta per le sottolineature, (Fctn) (U). Il listato 4 riporta un breve esempio di sottolineatura.

Veniamo ora ai caratteri in sovrascrittura, intendendo per questa, la stampa di caratteri più piccoli di quelli normali, nella metà più alta della riga, come accade per gli esponenti nelle formule matematiche. Il codice da inviare è Esc S + (n), dove n può assumere i valori 0 per la sovrascrittura e 1 per la sottoscrittura. In pratica bisogna scrivere: Print # 1:Chr$(27); "S"; Chr$(0); Chr$(15);. Quando si inviano questi codici vengono stampati tutti i caratteri presenti nel buffer di stampa e quelli che seguono vengono stampati come caratteri sovrascritti nella metà superiore della riga. La stampa avviene in modo unidirezionale e in modo doppio. Dopo un primo passaggio la carta viene fatta avanzare di 1/216 di pollice e la testina passa una seconda volta. A causa di questo avanzamento la stampante procede automaticamente all'aggiustamento dell'interlinea per mantenere la lunghezza assoluta del modulo. Il codice Esc T, cioè Chr$(27);"T";, annulla la sovrascrittura, ma non la stampa doppia. Per questo è necessario al termine disattivarla con il codice Esc H.

Il listato 5 offre come esempio una equazione di terzo grado. Si ricorda che i codici "H", "T", "S", ecc, possono esser sostituiti con i rispettivi numeri di codice ASCII, Chr$(72),Chr$(84),Chr$(83), ecc.

Non riteniamo con queste poche righe di aver esaurito un argomento che, come ben si sa, richiede un intero manuale; pensiamo tuttavia di aver toccato alcuni punti di un certo interesse che possono aprire la via ad un uso più ampio di questo indispensabile dispositivo che è la stampante, al di là del puro e semplice listato, alla quale viene spesso relegata.

Ci proponiamo, in un prossimo articolo, di sviluppare la parte che riguarda la stampa in Bit Image, cioè quella in modo grafico, alla quale, sia pur indirettamente abbiamo accennato nel numero di Marzo con il programma *Sub Copy* e che, come avrete modo di vedere, riserva ben altre sorprese. ■

Listato 5 - Prova di stampa in grassetto con apici e pedici.

```
100 OPEN #1:"RS232.DA=8"
110 PRINT #1:CHR$(27);"E";
120 PRINT #1:"Y=aX";CHR$(27);"F";
130 PRINT #1:CHR$(27);"S";CHR$(0);
140 PRINT #1:CHR$(15);"3";
150 PRINT #1:CHR$(27);"T";CHR$(18);
160 PRINT #1:CHR$(27);"E";
170 PRINT #1:CHR$(27);"H";
180 PRINT #1:"+bX";CHR$(27);"F";
190 PRINT #1:CHR$(27);"S";CHR$(0);
200 PRINT #1:CHR$(15);"2";
210 PRINT #1:CHR$(27);"T";CHR$(18);
220 PRINT #1:CHR$(27);"E";
230 PRINT #1:CHR$(27);"H";
240 PRINT #1:"+cX+d"
250 CLOSE #1
260 END
```

$Y=aX^3+bX^2+cX+d$

Bogey Boogie

Un piacevole brano musicale in onore di Humphrey Bogart

di Piero Chianura

L'esasperante lentezza nell'elaborazione dei dati, è nemica della realizzazione di programmi grafici e musicali con il TI99. Ma questo brano musicale sembra uscire da un computer ben più potente, tale è la sua velocità di esecuzione.

Caro Humphrey,

fra mille tributi possibili questo è, sicuramente, il più impensabile. Sono passati quasi trent'anni dalla tua scomparsa e di te ci si ricorda ancora: un cappello, un lungo impermeabile con il ba-

Listato 1 - L'allegro motivo in onore di Humphrey Bogart.

```
10 REM *** BOGEY   BOOGIE ***

20 REM *** Chianura Piero ***

30 CALL CLEAR
40 CALL SCREEN(2)
50 PRINT TAB(18);"PQRST":TAB(18)
;"U  VW":TAB(18);"X YZ.":TAB(18)
;";=8(9w":TAB(18);"xy )zt":TAB(1
8);"{v}~s":"qqqqqqqqqqqqqqqqqqq
qqqqqqqq"
60 PRINT "       BOGEY  BOOGIE":
:"   di Chianura Piero"::"qqqqqq
qqqqqqqqqqqqqqqqqqqq":::::
70 CALL CHAR(128,"0103071088473E
")
80 CALL CHAR(129,"FFE00F3FFFFF05
09")
90 CALL CHAR(130,"F83E83F0FCFFF9
E9")
100 CALL CHAR(131,"000080000080E
03")
110 CALL CHAR(132,"0000070003000
102")
120 CALL CHAR(133,"09088621F0FF0
F31")
130 CALL CHAR(134,"C9F102FC00FFA
C28")
140 CALL CHAR(135,"0C030C1")
150 CALL CHAR(136,"0000010000010
70C")
160 CALL CHAR(137,"1F7CC10F3FFF9
F9E")
170 CALL CHAR(138,"FF07F0FCFFFFA
09")
180 CALL CHAR(139,"80C0E00811E27
C")
190 CALL CHAR(140,"30C03008")
200 CALL CHAR(141,"9C8F403F00FF3
514")
210 CALL CHAR(142,"901061830FFFF
08C")
220 CALL CHAR(143,"0000E000C0008
04")
230 CALL CHAR(144,"0000000102040
81")
240 CALL CHAR(145,"1F20408")
250 CALL CHAR(146,"E0100C0201")

260 CALL CHAR(147,"0000000080402
01")
270 CALL CHAR(148,"0000000000001
82C")
280 CALL CHAR(149,"2040808080808
04")
290 CALL CHAR(150,"0809060001030
70F")
300 CALL CHAR(151,"DC3C7CF8F0E0C
08")
310 CALL CHAR(152,"2020201010080
804")
320 CALL CHAR(153,"00000000030F3
FFE")
330 CALL CHAR(154,"132140C0C08")

340 CALL CHAR(155,"0000E07030383
```

vero alzato ed una sigaretta accesa (in ombra, per giunta).
Meglio così! Che siano pochi intimi a ricordare la tua scarna bellezza ed il tuo mutevole comportamento, fra il romantico ed il cinico, l'insultante e l'acuto... Invano, alla continua ricerca di nuovi modelli, si guarda al passato. Spesso, si trova solo un abbigliamento.
Oggi, evidentemente, un mito non regge il confronto con i propri difetti e cade presto.

Che fortuna, per un uomo come te, essersi spogliato di impermeabile e cappello, aver gettato la sigaretta! La società del divismo ha raccolto e tenuto tutto. Ancora: che fortuna! Nel tremila, nessuno si ricorderà di te per criticarti; non sapranno mai se sotto il cappello eri pelato completamente, se sotto l'impermeabile eri nudo e se fumavi una sigaretta "strana"!

Analisi del listato

10-40 - Presentazione ed annerimento schermo.
50-60 - Prestampa Bogart e intestazione.
70-220 - Definizione caratteri disegno occhi.
230-520 - Definizione caratteri disegno Bogart.
530-540 - Schiarimento schermo.
550-720 - Stampa occhi.
730-1450 - Definizione valore delle note.
1460-1640 - Suona le parti 1-4.
1650-1820 - Suona le parti 5-7.
1830-1980 - Suona la parte 8.
1990-2160 - Suona le parti 9-11.
2170-2320 - Suona la parte 12.
2330-2490 - Suona la parte 13.
2500-2660 - Suona le parti 14-16.
2670-2720 - Suona la parte 17.
2730-2880 - Suona la parte 18.
2890-3050 - Suona la parte 19.
3060-3220 - Suona le parti 20-22.
3230-3280 - Suona la parte 23.
3290-3440 - Suona la parte 24.
3450-3600 - Suona la parte 25.
3610-3760 - Suona la parte 26.
3770-3920 - Suona la parte 27.
3930-3980 - Suona le parti 28-33.
3990-4140 - Suona la parte 34.
4150-4400 - Suona la parte 35 e torna da capo. ■

Seguito listato musicale.

```
838")
350 CALL CHAR(156,"04020C10237FF
FFF")
360 CALL CHAR(157,"030F3FFFFFCF
8E")
370 CALL CHAR(158,"FCF0E08")
380 CALL CHAR(40,"02010204080402
01")
390 CALL CHAR(159,"3C3E3F3F3F3F3
F3F")
400 CALL CHAR(119,"00000080C0E0F
0FF")
410 CALL CHAR(120,"7F")
420 CALL CHAR(121,"C0804020101C3
E7F")
430 CALL CHAR(41,"0201020462512A
14")
440 CALL CHAR(122,"3F3F3F3F3F3F3
F3F")
450 CALL CHAR(123,"0000010608102
04")
460 CALL CHAR(118,"7FFF7F7F7F7F7
F7F")
470 CALL CHAR(125,"80C0E0F0F8FFF
FFF")
480 CALL CHAR(126,"0C0000000102F
C")
490 CALL CHAR(127,"3F7F7FBF3F3F3
F3F")
500 CALL CHAR(115,"FFFFFFFFFFFF
FFF")
510 CALL CHAR(116,"FCFEFFFFFFFF
FFF")
520 CALL CHAR(113,"FF")
530 CALL COLOR(2,15,1)
540 CALL SCREEN(16)
550 R=5
560 CO=10
570 CALL HCHAR(R,CO,128)
580 CALL HCHAR(R,CO+1,129)
590 CALL HCHAR(R,CO+2,130)
600 CALL HCHAR(R,CO+3,131)
610 CALL HCHAR(R+1,CO,132)
620 CALL HCHAR(R+1,CO+1,133)
630 CALL HCHAR(R+1,CO+2,134)
640 CALL HCHAR(R+1,CO+3,135)
650 CALL HCHAR(R,CO+8,136)
660 CALL HCHAR(R,CO+9,137)
670 CALL HCHAR(R,CO+10,138)
680 CALL HCHAR(R,CO+11,139)
690 CALL HCHAR(R+1,CO+8,140)
700 CALL HCHAR(R+1,CO+9,141)
710 CALL HCHAR(R+1,CO+10,142)
720 CALL HCHAR(R+1,CO+11,143)
730 LAB=208
740 LGD=208
750 LG=196
760 LGB=185
```

Seguito listato musicale.

```
770 LFD=185
780 LF=175
790 LE=165
800 LEB=156
810 LDD=156
820 LD=147
830 LDB=139
840 LCD=139
850 LC=131
860 LLB=123
870 LLBB=117
880 LLAD=117
890 LLA=110
900 AB=415
910 GD=415
920 G=392
930 GB=370
940 FD=370
950 F=349
960 E=330
970 EB=311
980 DD=311
990 D=294
1000 DB=277
1010 CD=277
1020 C=262
1030 LB=247
1040 LBB=233
1050 LAD=233
1060 LA=220
1070 HAB=831
1080 HGD=831
1090 HG=784
1100 HGB=740
1110 HFD=740
1120 HF=698
1130 HE=659
1140 HEB=622
1150 HDD=622
1160 HD=587
1170 HDB=554
1180 HCD=554
1190 HC=523
1200 B=494
1210 BB=466
1220 AD=466
1230 A=440
```

```
1240 HHAB=1661
1250 HHGD=1661
1260 HHG=1568
1270 HHGB=1480
1280 HHFD=1480
1290 HHF=1397
1300 HHE=1319
1310 HHEB=1245
1320 HHDD=1245
1330 HHD=1175
1340 HHDB=1109
1350 HHCD=1109
1360 HHC=1047
1370 HB=988
1380 HBB=932
1390 HAD=932
1400 HA=880
1410 HHHC=2094
1420 HHB=1976
1430 HHBB=1864
1440 HHAD=1864
1450 HHA=1760
1460 T=-100
1470 FOR Z=1 TO 4
1480 CALL SOUND(T,LAB,0,HAB,0)
1490 CALL SOUND(T,LAB,0,AB,0)
1500 CALL SOUND(T,AB,0,HHAB,0)
1510 CALL SOUND(T,AB,0,HAB,0)
1520 CALL SOUND(T,LFD,0,HFD,0)
1530 CALL SOUND(T,LFD,0,FD,0)
1540 CALL SOUND(T,LEB,0,HEB,0)
1550 CALL SOUND(T,LEB,0,EB,0)
1560 CALL SOUND(T,LDD,0,HDD,0)
1570 CALL SOUND(T,LDD,0,DD,0)
1580 CALL SOUND(T,LDB,0,HDB,0)
1590 CALL SOUND(T,LDB,0,DB,0)
1600 CALL SOUND(T,LFD,0,HFD,0)
1610 CALL SOUND(T,LFD,0,FD,0)
1620 CALL SOUND(T,LDB,0,HDB,0)
1630 CALL SOUND(T,LDB,0,DB,0)
1640 NEXT Z
1650 FOR Z=1 TO 3
1660 CALL SOUND(T,HDB,0,LAB,0)
1670 CALL SOUND(T,HD,0,LAB,0)
1680 CALL SOUND(T,HEB,0,AB,0)
1690 CALL SOUND(T,HE,0,AB,0)
1700 CALL SOUND(T,HFD,0,LFD,0)
1710 CALL SOUND(T,HCD,0,LFD,0)
```

Seguito listato musicale.

```
1720 CALL SOUND(T,HDB,0,LEB,0)
1730 CALL SOUND(T,HDD,0,LEB,0)
1740 CALL SOUND(T,HDB,0,LDD,0)
1750 CALL SOUND(T,HDD,0,LDD,0)
1760 CALL SOUND(T,HEB,0,LDB,0)
1770 CALL SOUND(T,HE,0,LDB,0)
1780 CALL SOUND(T,HFD,0,LFD,0)
1790 CALL SOUND(T,HE,0,LFD,0)
1800 CALL SOUND(T,HEB,0,LDB,0)
1810 CALL SOUND(T,HDD,0,LDB,0)
1820 NEXT Z
1830 CALL SOUND(T,HDB,0,LAB,0,HA
B,0)
1840 CALL SOUND(T,LAB,0,AB,0)
1850 CALL SOUND(T,AB,0,HHAB,0)
1860 CALL SOUND(T,AB,0,HAB,0)
1870 CALL SOUND(T,LFD,0,HFD,0)
1880 CALL SOUND(T,LFD,0,FD,0)
1890 CALL SOUND(T,LEB,0,HEB,0)
1900 CALL SOUND(T,LEB,0,EB,0)
1910 CALL SOUND(T,LDD,0,HDD,0)
1920 CALL SOUND(T,LDD,0,DD,0)
1930 CALL SOUND(T,LDB,0,HHAB,0)

1940 CALL SOUND(T,LDB,0,HAB,0)
1950 CALL SOUND(T,LFD,0,HHAB,0)

1960 CALL SOUND(T,LFD,0,HAB,0)
1970 CALL SOUND(T,LDB,0,HCD,0)
1980 CALL SOUND(T,LDB,0,CD,0)
1990 FOR Z=1 TO 3
2000 CALL SOUND(T,HHAB,0,LEB,0)

2010 CALL SOUND(T,A,0,LEB,0)
2020 CALL SOUND(T,BB,0,EB,0)
2030 CALL SOUND(T,B,0,EB,0)
2040 CALL SOUND(T,HCD,0,LCD,0)
2050 CALL SOUND(T,HHG,0,LCD,0)
2060 CALL SOUND(T,HHAB,0,LLBB,0)

2070 CALL SOUND(T,AD,0,LLBB,0)
2080 CALL SOUND(T,HHAB,0,LLAD,0)

2090 CALL SOUND(T,AD,0,LLAD,0)
2100 CALL SOUND(T,BB,0,AB,0)
2110 CALL SOUND(T,B,0,AB,0)
2120 CALL SOUND(T,HCD,0,LAB,0)
2130 CALL SOUND(T,B,0,LAB,0)
2140 CALL SOUND(T,BB,0,AB,0)
2150 CALL SOUND(T,AD,0,AB,0)
2160 NEXT Z
2170 CALL SOUND(T,HHAB,0,LEB,0,H
EB,0)
2180 CALL SOUND(T,LEB,0,EB,0)
2190 CALL SOUND(T,EB,0,HHEB,0)
2200 CALL SOUND(T,EB,0,HEB,0)
2210 CALL SOUND(T,LCD,0,HCD,0)
2220 CALL SOUND(T,LCD,0,CD,0)
2230 CALL SOUND(T,LLBB,0,BB,0)
2240 CALL SOUND(T,LLBB,0,LBB,0)

2250 CALL SOUND(T,LLAD,0,AD,0)
2260 CALL SOUND(T,LLAD,0,LAD,0)

2270 CALL SOUND(T,AB,0,HEB,0)
2280 CALL SOUND(T,AB,0,EB,0)
2290 CALL SOUND(T,LAB,0,HCD,0)
2300 CALL SOUND(T,LAB,0,CD,0)
2310 CALL SOUND(T,AB,0,HHAB,0)
2320 CALL SOUND(T,AB,0,HAB,0)
2330 FOR Y=1 TO 2
2340 CALL SOUND(T,LDB,0)
2350 CALL SOUND(T,LDB,0,DB,0)
2360 CALL SOUND(T,DB,0,HEB,0,HHD
B,0)
2370 CALL SOUND(T,DB,0,HDB,0)
2380 CALL SOUND(T,LDB,0,HDB,0)
2390 CALL SOUND(T,LDB,0,DB,0)
2400 CALL SOUND(T,DB,0,BB,0)
2410 CALL SOUND(T,DB,0,LBB,0)
2420 CALL SOUND(T,LDB,0,AD,0)
2430 CALL SOUND(T,LDB,0,LAD,0)
2440 CALL SOUND(T,DB,0,HHGB,0)
2450 CALL SOUND(T,DB,0,HGB,0)
2460 CALL SOUND(T,LDB,0,HHF,0)
2470 CALL SOUND(T,LDB,0,HF,0)
2480 CALL SOUND(T,DB,0,HHDB,0)
2490 CALL SOUND(T,DB,0,HDB,0)
2500 CALL SOUND(T,LCD,0,A,0,HHCD
,0)
2510 CALL SOUND(T,LCD,0,CD,0)
2520 CALL SOUND(T,CD,0,HCD,0)
2530 CALL SOUND(T,HCD,0,HHCD,0)

2540 CALL SOUND(T,LCD,0,CD,0)
2550 CALL SOUND(T,CD,0,HCD,0)
```

Seguito listato musicale.

```
2560 CALL SOUND(T,CD,0,A,0,HHCD,
0)
2570 CALL SOUND(T,HCD,0,HHCD,0)

2580 CALL SOUND(T,LCD,0,CD,0)
2590 CALL SOUND(T,LCD,0,CD,0)
2600 CALL SOUND(T,CD,0,HCD,0)
2610 CALL SOUND(T,HCD,0,HHCD,0)

2620 CALL SOUND(T,LCD,0,A,0,HHCD
,0)
2630 CALL SOUND(T,LCD,0,CD,0)
2640 CALL SOUND(T,CD,0,HCD,0)
2650 CALL SOUND(T,HCD,0,HHCD,0)
2660 NEXT Y
2670 FOR Y=1 TO 4
2680 CALL SOUND(T,LAB,0,HHAB,0)

2690 CALL SOUND(T,LAB,0,HHG,0)
2700 CALL SOUND(T,AB,0,HHAB,0)
2710 CALL SOUND(T,AB,0,AD,0)
2720 NEXT Y
2730 CALL SOUND(T,LAB,0,HHAB,0)

2740 CALL SOUND(T,LAB,0,HHGD,0)

2750 CALL SOUND(T,AB,0,HHGB,0)
2760 CALL SOUND(T,AB,0,HHFD,0)
2770 CALL SOUND(T,LAB,0,HHF,0)
2780 CALL SOUND(T,LAB,0,HHFD,0)

2790 CALL SOUND(T,AB,0,HHGB,0)
2800 CALL SOUND(T,AB,0,HHGD,0)
2810 CALL SOUND(T,LAB,0,HHAB,0)

2820 CALL SOUND(T,LAB,0,HHGD,0)

2830 CALL SOUND(T,AB,0,HHGB,0)
2840 CALL SOUND(T,AB,0,HHFD,0)
2850 CALL SOUND(T,LAB,0,HHF,0)
2860 CALL SOUND(T,LAB,0,AB,0)
2870 CALL SOUND(T,AB,0,HAB,0)
2880 CALL SOUND(T,HAB,0,HHAB,0)

2890 FOR Y=1 TO 2
2900 CALL SOUND(T,LEB,0)
2910 CALL SOUND(T,LEB,0,EB,0)

2920 CALL SOUND(T,EB,0,HFD,0,HHE
B,0)
2930 CALL SOUND(T,EB,0,HEB,0)
2940 CALL SOUND(T,LEB,0,HEB,0)
2950 CALL SOUND(T,LEB,0,EB,0)
2960 CALL SOUND(T,EB,0,HCD,0)
2970 CALL SOUND(T,EB,0,CD,0)
2980 CALL SOUND(T,LEB,0,HC,0)
2990 CALL SOUND(T,LEB,0,C,0)
3000 CALL SOUND(T,EB,0,HHAB,0)
3010 CALL SOUND(T,EB,0,HAB,0)
3020 CALL SOUND(T,LEB,0,HHGB,0)

3030 CALL SOUND(T,LEB,0,HGB,0)
3040 CALL SOUND(T,EB,0,HHEB,0)
3050 CALL SOUND(T,EB,0,HEB,0)
3060 CALL SOUND(T,LD,0,HE,0,HHD,
0)
3070 CALL SOUND(T,LD,0,D,0)
3080 CALL SOUND(T,D,0,HD,0)
3090 CALL SOUND(T,HD,0,HHD,0)
3100 CALL SOUND(T,LD,0,D,0)
3110 CALL SOUND(T,D,0,HD,0)
3120 CALL SOUND(T,D,0,HE,0,HHD,0
)
3130 CALL SOUND(T,HD,0,HHD,0)
3140 CALL SOUND(T,LD,0,D,0)
3150 CALL SOUND(T,LD,0,D,0)
3160 CALL SOUND(T,D,0,HD,0)
3170 CALL SOUND(T,HD,0,HHD,0)
3180 CALL SOUND(T,LD,0,HE,0,HHD,
0)
3190 CALL SOUND(T,LD,0,D,0)
3200 CALL SOUND(T,D,0,HD,0)
3210 CALL SOUND(T,HD,0,HHD,0)
3220 NEXT Y
3230 FOR Y=1 TO 4
3240 CALL SOUND(T,LEB,0,HHEB,0)

3250 CALL SOUND(T,LEB,0,HHD,0)
3260 CALL SOUND(T,EB,0,HHEB,0)
3270 CALL SOUND(T,EB,0,HHF,0)
3280 NEXT Y
3290 CALL SOUND(T,LEB,0,HHEB,0)

3300 CALL SOUND(T,LEB,0,HHDD,0)

3310 CALL SOUND(T,EB,0,HHDB,0)
3320 CALL SOUND(T,EB,0,HHCD,0)
```

Seguito listato musicale.

```
3330 CALL SOUND(T,LEB,0,HHC,0)
3340 CALL SOUND(T,LEB,0,HHCD,0)

3350 CALL SOUND(T,EB,0,HHDB,0)
3360 CALL SOUND(T,EB,0,HHDD,0)
3370 CALL SOUND(T,LEB,0,HHEB,0)

3380 CALL SOUND(T,LEB,0,HHDD,0)

3390 CALL SOUND(T,EB,0,HHDB,0)
3400 CALL SOUND(T,EB,0,HHCD,0)
3410 CALL SOUND(T,LEB,0,HHC,0)
3420 CALL SOUND(T,LEB,0,EB,0)
3430 CALL SOUND(T,EB,0,HEB,0)
3440 CALL SOUND(T,HEB,0,HHEB,0)

3450 CALL SOUND(T,LLBB,0,HHEB,0)

3460 CALL SOUND(T,LLBB,0,HHE,0)

3470 CALL SOUND(T,LBB,0,HHFD,0)

3480 CALL SOUND(T,LBB,0,HHGB,0)

3490 CALL SOUND(T,LLBB,0,HHG,0)

3500 CALL SOUND(T,LLBB,0,HHD,0)
3510 CALL SOUND(T,LBB,0,HHEB,0)

3520 CALL SOUND(T,LBB,0,HHF,0)
3530 CALL SOUND(T,LLBB,0,HHEB,0)

3540 CALL SOUND(T,LLBB,0,HHF,0)

3550 CALL SOUND(T,LBB,0,HHFD,0)

3560 CALL SOUND(T,LBB,0,HHGB,0)

3570 CALL SOUND(T,LLBB,0,HHG,0)

3580 CALL SOUND(T,LLBB,0,HHGB,0)

3590 CALL SOUND(T,LBB,0,HHFD,0)

3600 CALL SOUND(T,LBB,0,HHF,0)
3610 CALL SOUND(T,LLBB,0,HHEB,0)

3620 CALL SOUND(T,LLBB,0,HHE,0)

3630 CALL SOUND(T,LBB,0,HHFD,0)

3640 CALL SOUND(T,LBB,0,HHGB,0)

3650 CALL SOUND(T,LLBB,0,HHFD,0)

3660 CALL SOUND(T,LLBB,0,HHGB,0)

3670 CALL SOUND(T,LBB,0,HHG,0)
3680 CALL SOUND(T,LBB,0,HHAB,0)

3690 CALL SOUND(T,LLBB,0,HHG,0)

3700 CALL SOUND(T,LLBB,0,HHGB,0)

3710 CALL SOUND(T,LBB,0,HHFD,0)

3720 CALL SOUND(T,LBB,0,HHF,0)
3730 CALL SOUND(T,LLBB,0,HHEB,0)

3740 CALL SOUND(T,LLBB,0,HHD,0)

3750 CALL SOUND(T,LBB,0,HHDB,0)

3760 CALL SOUND(T,LBB,0,HHCD,0)

3770 CALL SOUND(T,LDB,0,HHDB,0)

3780 CALL SOUND(T,LDB,0,HHD,0)
3790 CALL SOUND(T,DB,0,HHEB,0)
3800 CALL SOUND(T,DB,0,HHF,0)
3810 CALL SOUND(T,LDB,0,HHEB,0)
3820 CALL SOUND(T,LDB,0,HHF,0)
3830 CALL SOUND(T,DB,0,HHFD,0)
3840 CALL SOUND(T,DB,0,HHGB,0)
3850 CALL SOUND(T,LDB,0,HHFD,0)

3860 CALL SOUND(T,LDB,0,HHGB,0)

3870 CALL SOUND(T,DB,0,HHG,0)
3880 CALL SOUND(T,DB,0,HHAB,0)
3890 CALL SOUND(T,LDB,0,AD,0)
3900 CALL SOUND(T,LDB,0,BB,0)
3910 CALL SOUND(T,DB,0,HC,0)
3920 CALL SOUND(T,DB,0,HCD,0)
3930 FOR Y=1 TO 8
```

Seguito listato musicale.

```
3940 CALL SOUND(T,LDB,0,HDB,0)
3950 CALL SOUND(T,LDB,0,HCD,0)
3960 CALL SOUND(T,DB,0,HDB,0)
3970 CALL SOUND(T,DB,0,HDD,0)
3980 NEXT Y
3990 CALL SOUND(T,LDB,0,HDB,0,HH
DB,0)
4000 CALL SOUND(T,LDB,0,DB,0,HDB
,0)
4010 CALL SOUND(T,HDB,0,HHDB,0,L
DB,0)
4020 CALL SOUND(T,HDB,0,DB,0,LDB
,0)
4030 CALL SOUND(T,HDD,0,HHDD,0,L
DD,0)
4040 CALL SOUND(T,HDD,0,DD,0,LDD
,0)
4050 CALL SOUND(T,HDB,0,HHDB,0,L
EB,0)
4060 CALL SOUND(T,HDB,0,DB,0,LEB
,0)
4070 CALL SOUND(T,HCD,0,HHCD,0,L
E,0)
4080 CALL SOUND(T,CD,0,HCD,0,LE,
0)
4090 CALL SOUND(T,HC,0,HHC,0,LFD
,0)
4100 CALL SOUND(T,C,0,HC,0,LFD,0
)
4110 CALL SOUND(T,BB,0,HBB,0,LGB
,0)
4120 CALL SOUND(T,LBB,0,BB,0,LGB
,0)
4130 CALL SOUND(T,AD,0,HAD,0,LG,
0)
4140 CALL SOUND(T,LAD,0,AD,0,LG,
0)
4150 CALL SOUND(T,HHAB,0,HHHC,0,
LAB,0)
4160 CALL SOUND(T,LAB,0,AB,0)
4170 CALL SOUND(T,AB,0,HHAB,0)
4180 CALL SOUND(T,AB,0,HAB,0)
4190 CALL SOUND(T,LFD,0,HFD,0)
4200 CALL SOUND(T,LFD,0,FD,0)
4210 CALL SOUND(T,LEB,0,HEB,0)
4220 CALL SOUND(T,LEB,0,EB,0)
4230 CALL SOUND(T,LDD,0,HDD,0)
4240 CALL SOUND(T,LDD,0,DD,0)
4250 IF Z=6 THEN 4340
4260 CALL SOUND(T,LDB,0,HDB,0)
4270 CALL SOUND(T,LDB,0,DB,0)
4280 CALL SOUND(T,LFD,0,HFD,0)
4290 CALL SOUND(T,LFD,0,FD,0)
4300 CALL SOUND(T,LDB,0,HDB,0)
4310 CALL SOUND(T,LDB,0,DB,0)
4320 Z=6
4330 GOTO 4150
4340 CALL SOUND(T,LDB,0,HHDB,0)
4350 CALL SOUND(T,LDB,0,HDB,0)
4360 CALL SOUND(T,LFD,0,HHDB,0)
4370 CALL SOUND(T,LFD,0,HDB,0)
4380 CALL SOUND(T,LDB,0,HHFD,0)
4390 CALL SOUND(T,LDB,0,HFD,0)
4400 GOTO 1650
```

Forza 4

Anche per il computer Texas uno dei più famosi giochi di filetto

di Sergio Borsani

Forza 4 si inserisce nell'insieme dei giochi del filetto tra i quali si annovera anche la classica Tria. Tutti hanno come denominatore comune il fatto che per vincere bisogna riuscire a mettere in fila un certo numero di simboli, pedine o contrassegni. Forza 4 differisce dal cugino filetto per la caratteristica di presentare un campo di battaglia composto da sette caselle per sei, in posizione verticale per cui, scelta la colonna, si va sempre a occupare la casella libera più in basso, non potendo le pedine sostare a mezz'aria.

La trasposizione del gioco al computer ha un suo fascino particolare, anche se non come quello riscontrabile tra due persone, ma piuttosto per lo sforzo progettuale e creativo che implica la codifica delle vie attraverso le quali si esplica la strategia.

A questo riguardo ci si può sbizzarrire. Ci è noto un tentativo di un programma di autoapprendimento per Apple che imparasse a giocare a filetto su una base di tre caselle per tre. In pratica il computer, che inizialmente giocava in modo pessimo, in seguito, distingueva la bontà delle proprie mosse in base agli esiti delle partite precedenti memorizzava il tutto e poi evitava accuratamente quelle mosse che avevano condotto alla sconfitta.

Il metodo, che ricorda da vicino il nostro modo di apprendere per tentativi ed errori, è interessante, ma non crediamo che sia il più conveniente per risolvere il problema. Generalmente, in questi casi, si preferisce far in modo che il programma possa valutare la situazione che verrebbe a crearsi dopo una o più mosse. I criteri per l'elaborazione vengono così fossilizzati in una serie di procedure e di controlli espressi in una forma generale e tali da poter essere applicati alle diverse situazioni che vengono a crearsi sulla scacchiera.

Un aspetto che non va trascurato è il tempo che impiega il computer a elaborare la risposta. Un Chess challenger al livello cinque impiega anche 15 minuti per mossa; ma mentre negli scacchi ciò è del tutto naturale, ve lo immaginate il

Listato 1 - Il programma Forza 4.

```
100 !      F O R Z A - 4
110 !
120 !    di Sergio Borsani
130 !versione Extended BASIC
140 DIM F$(24),Q(7):: CALL CLEAR :: RAND
OMIZE :: FOR I=0 TO 24 :: F$(I)=RPT$(" "
,9):: NEXT I :: GOTO 270
150 DATA "000103030707070F","FFFFFFFFFFF
FF0E","E3F7F7F7F7EF1FFF","F0F8FCFCFEFEFE
FF","1F3F7F7FFFF8F07"
160 DATA "E0F0F8FCFCFE7E7E","7FFFFFFFF7
F0703","FCFEFEFEFCFCF8F","070F1F1F3F3F7F
7F","80C0C0E0E0E0E0F"
170 DATA "0F0F1F1F1F3F3F3F","E1F1FEFFFFF
FFFFE","FCF8F870707070F8","7F7F3F3F3F3F7
FFE","606060717F7F7FFF"
180 DATA "7E7EFCFCF8F0E0E","070F0F1F3F7F
FFFF","F0E0C1C183830707","FFFDF8F8F8F0F0
FF","F0F0F8F8F8F8FCFC"
190 DATA "7F7F7FFFFFFF7F3E","8103030101"
,"F9FFFFFFFFFF7F1F","FEFEFDFBF7F7EF87","
E7E3C3C1C1808"
200 DATA "E1F3FBFDFDFEFE7D","FEFEFFFFFFF
FFFFF","0F0FF7FBFBFBFBF7","FFFFE0C0C0808
","FCFEFE7F7F3F3F1E"
210 DATA "00007F7FFEFE","0","03070F0F1F3
F7FFB","F0F8F8F8F8F8F8F8","010307070F1F3
F3F","F3F3E3E3C7C7FFFF"
220 DATA "F8F8F8F8F8FFFFFF","3F1F","FFFF
1F0F0F0F0F07","FEFEFEF0F0F0F0E","384","3
0408","3C608"
230 N,J,R,C,TM,LV,G,X,XS,Y,K,S,P,XW,X0,M
,I,D,E,R1,R2,C1,C2,=0 :: E$,R$,SC$,A$,W$
,RR$,=""
240 CALL CHAR :: CALL COLOR :: CALL SCRE
EN :: CALL HCHAR :: CALL VCHAR :: CALL S
PRITE :: CALL SOUND
250 CALL DELSPRITE :: CALL CLEAR :: CALL
 MAGNIFY :: CALL LOCATE :: CALL KEY
260 !@P-
270 FOR N=96 TO 138 :: READ E$ :: CALL C
HAR(N,E$):: NEXT N :: FOR J=9 TO 14 :: C
ALL COLOR(J,4,4):: NEXT J :: CALL SCREEN
(4)
280 N=96 :: FOR R=10 TO 12 :: FOR C=10 T
O 19 :: CALL HCHAR(R,C,N):: N=N+1 :: NEX
T C :: NEXT R
290 CALL HCHAR(11,20,126):: CALL HCHAR(1
0,22,128):: CALL HCHAR(10,23,129):: N=13
0 :: FOR R=11 TO 12 :: FOR C=21 TO 23 ::
 CALL HCHAR(R,C,N)
300 N=N+1 :: NEXT C :: NEXT R :: FOR J=9
 TO 12 :: CALL COLOR(J,5,4):: NEXT J ::
CALL COLOR(13,13,4)
310 CALL SPRITE(#1,136,16,74,121,#2,136,
16,74,174,#3,137,16,74,140,#4,138,16,74,
80):: DISPLAY AT(20,6):"DI SERGIO BORSAN
I."
```

Seguito listato Forza 4.

```
320 CALL SOUND(150,659,2):: CALL SOUND(1
50,587,2):: CALL SOUND(300,523,2):: CALL
 SOUND(300,440,2):: CALL SOUND(300,698,2
,440,5)
330 CALL SOUND(300,784,2,587,5):: CALL S
OUND(150,698,2,392,5):: CALL SOUND(150,6
59,2):: CALL SOUND(1000,698,2,440,5)
340 FOR TM=1 TO 1000 :: NEXT TM :: CALL
DELSPRITE(ALL):: CALL CLEAR :: DISPLAY A
T(1,1):"PER ESEGUIRE UNA MOSSA USA I"
350 DISPLAY AT(2,1):"TASTI CON LE FRECCE
" :: DISPLAY AT(3,1):"POI PREMI IL TASTO
 <ENTER>." :: DISPLAY AT(5,1):"VUOI LA P
RIMA MOSSA? (Y/N) Y"
360 CALL COLOR(12,13,4,13,5,4,14,9,4)::
CALL MAGNIFY(3):: ACCEPT AT(5,28)VALIDAT
E("YN")SIZE(-1):R$
370 RESTORE 380 :: FOR J=128 TO 131 :: R
EAD E$ :: CALL CHAR(J,E$):: CALL CHAR(J+
8,E$):: NEXT J :: CALL CHAR(127,RPT$("F"
,16))
380 DATA "00071F3F3F7F7F7F","7F7F7F3F3F1
F0700","00E0F8FCFCFEFEFE","FEFEFEFCFCF8E
000"
390 DISPLAY AT(10,1):"SCEGLI IL LIVELLO
DI GIOCO" :: DISPLAY AT(12,8):"<1>   <2>
   <3>" :: DISPLAY AT(14,1)BEEP:"SCELTA:
"
400 ACCEPT AT(14,9)VALIDATE("123")SIZE(1
):SC$ :: LV=VAL(SC$):: CALL CLEAR
410 FOR J=5 TO 23 STEP 3 :: CALL HCHAR(J
,6,127,22):: NEXT J :: FOR J=6 TO 27 STE
P 3 :: CALL VCHAR(5,J,127,19):: NEXT J
420 IF R$="N" THEN G=-1 :: X=INT(RND*3)+
3 ELSE 460
430 DISPLAY AT(1,4)BEEP:"ECCO LA MIA MOS
SA" :: CALL SPRITE(#1,128,9,15,121):: XS
=X*24+25 :: CALL LOCATE(#1,15,XS)
440 FOR TM=1 TO 200 :: NEXT TM :: A$="A"
 :: Q(X)=Q(X)+1 :: Y=Q(X):: GOSUB 900 ::
 GOSUB 960
450 ! MUOVE L'AVVERSARIO
460 IF M=42 THEN 840 ELSE G=1 :: DISPLAY
 AT(1,4)BEEP:"ESEGUI LA TUA MOSSA" :: CA
LL SPRITE(#1,136,5,15,121):: X=4
470 CALL KEY(0,K,S):: IF S=0 THEN 470
480 IF K=83 AND X>1 THEN X=X-1 :: GOTO 5
10
490 IF K=68 AND X<7 THEN X=X+1 :: GOTO 5
10
500 IF K=13 AND Q(X)<6 THEN 520 ELSE 470
510 XS=X*24+25 :: CALL LOCATE(#1,15,XS):
: GOTO 470
520 A$="B" :: Q(X)=Q(X)+1 :: Y=Q(X):: GO
SUB 900 :: GOSUB 960 :: W$="BBBB" :: GOS
UB 990
530 IF P THEN DISPLAY AT(1,4)BEEP:"HAI V
INTO!" :: GOTO 800
540 DISPLAY AT(1,4):"UN MOMENTO, PREGO."
 :: K=1 :: A$="A" :: W$="AAAA" :: G=-1 :
: IF M=42 THEN 840
550 ! MUOVE IL COMPUTER
560 IF M<41 THEN 590
570 FOR J=1 TO 7 :: IF Q(J)<6 THEN 430
580 NEXT J :: GOTO 840
590 FOR X=1 TO 7 :: Y=Q(X)+1 :: IF Y>6 T
HEN 630 ELSE GOSUB 900 :: GOSUB 990
600 IF P AND K=1 THEN Q(X)=Q(X)+1 :: GOS
UB 960 :: DISPLAY AT(1,4):"MI DISPIACE,
HAI PERSO." :: GOSUB 1050 :: GOTO 850
610 IF P AND K=2 THEN A$="A" :: GOTO 430
620 A$=" " :: GOSUB 900 :: IF K=1 THEN A
$="A" ELSE A$="B"
630 NEXT X :: IF K=1 THEN K=2 :: A$="B"
:: W$="BBBB" :: GOTO 590
640 X=INT(RND*4):: X=INT(RND*4)+X+1 :: I
F Q(X)=6 THEN 640 ELSE XW=X
650 IF LV=1 THEN 430 ELSE A$="B"
660 FOR X=1 TO 7 :: Y=Q(X)+1 :: IF Y>6 T
HEN 720 ELSE GOSUB 900
670 P=POS(F$(7),"BBB",2):: IF P=0 OR P=2
 OR P=6 THEN 690
680 IF SEG$(F$(7),P-1,1)=" " AND SEG$(F$
(7),P+3,1)=" " THEN 430
690 FOR J=8 TO 12 :: P=POS(F$(J),"BBB",2
):: IF P=0 OR P=2 OR P=6 THEN 710
700 IF SEG$(F$(J),P-1,1)=" " AND SEG$(F$
(J),P+3,1)=" " AND SEG$(F$(J-1),P-1,1)<>
" " AND SEG$(F$(J),P+3,1)<>" " THEN 430
710 NEXT J :: A$=" " :: GOSUB 900 :: A$=
"B"
720 NEXT X :: IF LV=2 THEN X=XW :: GOTO
430
730 X=XW :: IF M>40 THEN 430 ELSE A$="A"
 :: GOSUB 900 :: XO=X
740 A$="B" :: W$="BBBB" :: FOR X=1 TO 7
:: Y=Q(X)+1 :: IF Y>6 THEN 790 ELSE GOSU
B 900 :: GOSUB 990
750 IF P=0 THEN 780
760 XW=XW+1 :: IF XW=XO THEN X=XW :: GOT
O 430 ELSE IF XW>7 THEN XW=1
770 A$="A" :: X=XW :: Y=Q(X)+1 :: GOSUB
900 :: GOTO 740
780 A$=" " :: GOSUB 900 :: A$="B"
790 NEXT X :: X=XW :: GOTO 430
800 RESTORE 810 :: FOR J=1 TO 27 :: READ
 D,E :: CALL SOUND(D,E,0):: NEXT J :: GO
TO 850
810 DATA 150,294,150,330,150,349,150,392
,150,440,150,440,300,440,150,523,150,406
,150,440,150,392
820 DATA 300,440,300,440,150,466,150,392
,150,392,150,392,150,440,150,349,150,149
,150,349,150,440
830 DATA 150,392,150,349,150,330,300,294
,300,294
840 DISPLAY AT(1,4):"GIOCO PARI."
850 DISPLAY AT(3,2)BEEP:"VUOI RICOMINCIA
RE? (Y/N) Y" :: ACCEPT AT(3,27)VALIDATE(
"YN")SIZE(-1):RR$ :: IF RR$="N" THEN CAL
L CLEAR :: END
860 M=0 :: FOR J=1 TO 7 :: Q(J)=0 :: NEX
T J :: FOR J=0 TO 24 :: F$(J)=RPT$(" ",9
```

gioco del filetto condotto su questi ritmi? Il Basic purtroppo non è il linguaggio più adatto per risolvere il problema della velocità. Prendiamo a esempio proprio il programma presentato in queste pagine: una sua prima versione per Olivetti M20 aveva un tempo medio di risposta attorno ai 30 secondi. Migliorando la routine per memorizzare le mosse e per cercare la combinazione vincente, si è passati, per un livello di gioco equivalente, a 27 secondi sul TI99.

Tenuto conto della diversa velocità dei due computer, si può parlare di un'efficienza tre o quattro volte superiore. Per memorizzare le mosse eseguite, verrebbe spontaneo utilizzare una matrice a due dimensioni e associare a ogni suo elemento un simbolo a seconda che la mossa sia stata effettuata dal computer o dal concorrente. Poiché la parte usata con maggiore frequenza è la ricerca nella tabella, è preferibile velocizzare al massimo la routine che la riguarda, anche a scapito delle altre.

Nel programma, lo schema della partita viene riprodotto sul vettore unidimensionale F$() con 25 elementi, ognuno dei quali rappresenta non una casella, ma un'intera riga, colonna e diagonale. Gli elementi con indice da zero a sei sono associati alle colonne, quelli con indice da sette a 12 alle righe, quelli con indice da 13 a 18 alle diagonali con inclinazione a sinistra e infine quelli con indice da 19 a 24 alle diagonali con inclinazione a destra. Per ogni mossa una pedina va a occupare una casella di coordinate X, Y; delle formule matematiche modificano gli elementi del vettore F$() corrispondenti alla riga, alla colonna e alle diagonali che passano per X, Y. Se la memorizzazione è un po' complicata, la ricerca avviene in modo molto spedito.

Simulata una mossa X, Y, con la funzione Pos si può immediatamente verificare se nella riga, nella colonna o nelle diagonali interessate si è venuta a creare la configurazione di simboli "AAAA" oppure "BBBB" che indicano un filetto ottenuto dal computer o dall'avversario.

La strategia insita nel programma non è in realtà molto sofisticata, nel senso che il computer invece di costruire la propria vittoria, pensa piuttosto a evitare la sconfitta. In gran parte ciò è dovuto al contenimento dei tempi di risposta. L'esame in profondità delle mosse e delle possibili risposte dell'avversario comporta la crescita in maniera esponenziale delle operazioni richieste. Non è però escluso che qualcuno voglia aggiungere una routine per elevare il livello del gioco.

La versione attuale presenta tre livelli di gioco. Per prima cosa il computer verifica la possibilità immediata di ottenere il filetto, in secondo luogo cerca se tale possibilità è a portata di mano dell'avversario, nel qual caso impedirà che ciò avvenga. Un altro controllo riguarda l'eventualità da parte dell'avversario di ottenere tre simboli contigui (BBB) con possibilità di filetto su entrambi i lati. Un ulteriore passo consiste nel verificare che una mossa non consenta all'avversario di ottenere filetto, ponendo la propria pedina sopra l'ultima collocata dal computer. Fatti salvi i controlli previsti dal livello di gioco scelto inizialmente, il computer elabora una mossa in modo casuale in modo che una certa situazione non conduca sempre alla medesima risposta. Con uno stratagemma di natura matematica si dà una maggiore probabilità alle colonne centrali; invece di ottenere un numero casuale da uno a sette, si sommano due numeri casuali da zero a tre e poi si aggiunge uno. In tal modo la quarta colonna viene scelta con più probabilità.

All'apertura del programma, dopo il titolo, si decide chi deve iniziare e si sceglie il livello di gioco. Appare poi lo schema del gioco: 42 caselle disposte su sei righe e sette colonne. Per eseguire la mossa si colloca la pedina in corrispondenza di una colonna usando i tasti con le frecce, <S> e <D>, senza premere il tasto Fctn, confermando poi con il tasto Enter. Accertatevi che il tasto Alpha Lock sia in posizione abbassata.

La partita viene accompagnata da messaggi che appaiono nella parte alta dello schermo. Vince chi per primo riesce a disporre quattro pedine contigue sulla stessa riga, colonna o diagonale. Buon divertimento e un ultimo suggerimento: provate a sostituire alle linee di programma che accettano la mossa del concorrente, un sottoprogramma in grado di competere con il programma principale. Assisterete a una lotta condotta esclusivamente a colpi di bit.

REMarks

100-260 - Dimensionamento delle matrici, definizione dei caratteri per il titolo e dichiarazione delle variabili e dei sottoprogrammi prima della sospensione del pre-scan.
270-330 - Scrittura del titolo.
340-400 - Scelta del livello di gioco.
410-420 - Disegno dello schema sul quale si svolge il gioco.
430-440 - Esecuzione della mossa da parte del computer.
450-530 - Indicazione della mossa da parte del concorrente.
540-790 - Elaborazione della mossa da parte del computer.
540-650 - Livello uno.
660-720 - Livello due.
730-790 - Livello tre.
800-830 - Musica suonata in caso di vittoria.
840 - Messaggio di parità.
850-880 - Richiesta di eventuale prosecuzione del gioco.
890-940 - Subroutine per memorizzare una mossa.
950-970 - Subroutine per visualizzare una mossa sullo schermo.
980-1040 - Subroutine per la ricerca di una data sequenza di mosse tra quelle memorizzate.
1050-1060 - Musica suonata in caso di perdita da parte del concorrente.

Lista delle variabili

Variabili stringa
F$() - Matrice per memorizzare la configurazione di simboli presenti in ogni riga, colonna e diagonale dello schema.
E$ - Variabile usata per leggere nelle istruzioni Data le stringhe esadecimali che definiscono i caratteri che formano il titolo.
R$ - Caricata con i valori "Y" o "N", indica se in apertura del gioco la prima mossa sia toccata al computer o al concorrente.
SC$,RR$ - Accettano le risposte nelle istruzioni Accept.
A$ - Contiene il simbolo da memorizzare in seguito a una mossa, "A" per il computer, "B" per il concorrente.
**W$** - Rappresenta la configurazione di simboli da cercare nella matrice F$().

Variabili numeriche

Q() - Matrice per memorizzare il numero di simboli già presentati in ogni colonna dello schema.

N,J - Usate prevalentemente nei cicli Fort-Next.

R,C - Indicano il numero di riga e di colonna nelle istruzioni Call Hchar.

TM - Variabile di controllo dei cicli per creare delle pause.

LV - Livello di gioco.

G - Assume i valori +1 e –1 in relazione a chi spetta la mossa.

X - Indica il numero di colonna per specificare la mossa da giocare.

XS - Esprime il valore di X in pixel, per posizionare lo sprite.

Y - Indica il numero di simboli già presenti nella colonna scelta per effettuare la mossa.

K,S - Variabili usate nell'istruzione Call Key; K contiene il numero di codice Ascii del tasto premuto, S esprime lo stato della tastiera.

P - Indica la presenza di una stringa all'interno di un'altra stringa. Se P è uguale a zero, la stringa non contiene i caratteri richiesti.

XW,XO - Memorizzano il numero di una colonna dello schema, durante i controlli per la ricerca della mossa elaborata dal programma.

M - Numero complessivo di mosse giocate.

I - Usata come indice della matrice F$().

D,E - Usate per leggere i dati da inserire nelle istruzioni Call Sound.

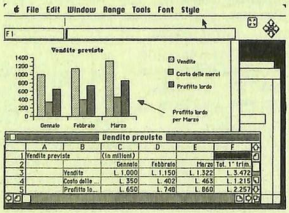
R1,R2,C1,C2 - Coordinate per collocare un simbolo nello schema, in seguito a una mossa.

Seguito listato Forza 4.

```
):: NEXT J :: IF R$="Y" THEN R$="N" ELSE
 R$="Y"
870 CALL CLEAR :: GOTO 390
880 END
890 ! MEMORIZZAZIONE MOSSA
900 I=X-1 :: GOSUB 910 :: I=Y+6 :: F$(I)
=SEG$(F$(I),1,X)&A$&SEG$(F$(I),X+2,8-X):
: GOTO 920
910 F$(I)=SEG$(F$(I),1,Y)&A$&SEG$(F$(I),
Y+2,8-Y):: RETURN
920 IF ABS(X+Y-7.5)<3 THEN I=X+Y+8 :: GO
SUB 910
930 IF ABS(Y-X+.5)<3 THEN I=Y-X+22 :: GO
SUB 910
940 RETURN
950 ! ESECUZIONE MOSSA
960 R1=(8-Y)*3 :: C1=X*3+4 :: R2=(R1-1)*
8+1 :: C2=(C1-1)*8+1 :: CALL LOCATE(#1,R
2,C2):: IF G=1 THEN N=128 ELSE N=136
970 FOR C=C1 TO C1+1 :: FOR R=R1 TO R1+1
 :: CALL VCHAR(R,C,N):: N=N+1 :: NEXT R
:: NEXT C :: CALL DELSPRITE(ALL):: M=M+1
 :: RETURN
980 ! RICERCA
990 P=POS(F$(X-1),W$,1):: IF P THEN RETU
RN
1000 P=POS(F$(Y+6),W$,1):: IF P THEN RET
URN
1010 IF ABS(X+Y-7.5)>=3 THEN 1040
1020 P=POS(F$(X+Y+8),W$,1):: IF P THEN R
ETURN
1030 IF ABS(Y-X+.5)>=3 THEN RETURN
1040 P=POS(F$(Y-X+22),W$,1):: RETURN
1050 CALL SOUND(750,175,1,349,1):: CALL
SOUND(700,220,1,440,1):: CALL SOUND(300,
208,1,415,1):: CALL SOUND(300,247,1,494,
1)
1060 CALL SOUND(800,175,1,349,1):: RETUR
N
```

Jazz su Macintosh: che musica!

Come certo i nostri lettori sapranno la Lotus deve le sue fortune, oltre ad un'azzeccata campagna pubblicitaria, all'"invenzione" del package integrato: quell'1-2-3 che da anni capeggia le classifiche di vendita del software standard negli Stati Uniti. La stessa filosofia è stata applicata anche per realizzare Symphony, progetto più ambizioso e meno fortunato del predecessore. La sintesi tra questo modo di pensare e l'ambiente a dialogo spontaneo del Macintosh (mouse, icone e menu), ha permesso la realizzazione di Jazz, package integrato ricco di sei programmi applicativi, facile da usare e da apprendere. I solisti sono: un foglio elettronico composto da una tabella di 256 colonne per 8.192 righe da gestire con il mouse; il programma per la grafica



che permette di tracciare diversi tipi di diagrammi a partire dal contenuto delle celle appartenenti al foglio di lavoro; il data base per l'archiviazione di dati; Forms per introdurre e richiedere informazioni al data base mediante pratiche maschere video; un elaboratore di testi che consente di correlare lettere e tabelle, circolari ed elenchi di indirizzi, grafici e commenti scritti; un programma di comunicazione che comprende i più diffusi standard e i protocolli per il dialogo PC/mainframe. Lotus Jazz dispone di un valido help on-line ed è distribuito in Italia da J.*soft*.

J.soft
Viale Restelli, 5
20124 Milano
Tel. 02-683797

I SEGRETI DEI PERSONAL

di Sergio Borsani

Input function

Un giorno, tra amici, si discuteva sulla possibilità di ridefinire una funzione matematica con una formula immessa in input da un programma Basic. Come si sa, in questi casi, generalmente si deve interrompere l'esecuzione del programma, editare la linea contenente l'istruzione Def e dare nuovamente il Run. La procedura non è elegante, ma, in mancanza d'altro, diventa accettabile. Ovviamente non funziona quello che potrebbe sembrare l'uovo di Colombo: accettare la formula in input con la stringa A$ e poi scrivere Def Y =A$.

Il programma che segue riesce nell'intento modificando i byte che codificano la linea di programma contenente l'istruzione Def mediante delle Call Peek e Call Load. Per il suo funzionamento è richiesto l'Extended Basic e l'espansione di memoria; quest'ultima non perché il programma sia troppo voluminoso, ma perché con tale configurazione il programma invece di risiedere nella Vdp Ram viene trasferito nella high memory dove può essere facilmente manipolato.

Alla linea 100 viene definita la funzione Y=X*2; la linea 110 legge il valore dei puntatori alla Tavola dei numeri di linea, mentre le seguenti calcolano l'indirizzo di memoria della linea 100. Il ciclo alla linea 140 legge e stampa in forma decimale i byte che costituiscono la linea 100.

Se viene dato il Run, sullo schermo appaiono i nove numeri riportati al termine del listato. 137 è il token della parola riservata Def, 89 è il codice Ascii della lettera Y, 190 è il token del segno =, 88 corrisponde alla X, 195 al segno di moltiplicazione, seguono i codici 200 e 1 e infine 50 che è il codice Ascii del numero due. L'ultimo byte della linea è sempe impostato a zero, mentre all'indirizzo immediatamente precedente la linea stessa c'è un byte che ne indica la lunghezza. Il codice 200 precede sempre una costante, il valore uno che lo segue sta a indicare che tale costante è formata da una sola cifra. A esempio il numero dieci verrebbe codificato con la sequenza 200, 2, 49, 48. Anche il punto decimale viene considerato alla stregua delle cifre.

È istruttivo cambiare la funzione presente nella linea 100, dare il Run e ottenere l'immagine della linea stessa così come si trova in memoria. Nel listato 2 la funzione è stata cambiata in Y=Sin(X).

Noti i valori token delle varie funzioni matematiche: Int, Sin, Log, eccetera, si può usare l'istruzione Call Load per intervenire direttamente nella Ram. Ma attenzione, se la nuova funzione occupa più byte della precedente c'è il rischio di sovrapporre due linee del programma. Questo non succede durante l'editing usuale poiché in tal caso il sistema operativo si incarica di cambiare gli indirizzi e di spostare tutta la tavola dei numeri di linea. La figura 1 illustra cosa avviene quando si edita la linea 100 scivendo una nuova funzione che occupa un byte di memoria in meno della precedente. Come si vede è cambiato l'indirizzo di linea ed è stata spostata di una locazione tutta la Tavola dei numeri di linea.

Il programma Basic in grado di emulare sotto questo aspetto il sistema operativo risulterebbe troppo lento, ma c'è il modo di aggirare l'ostacolo. La prima volta si

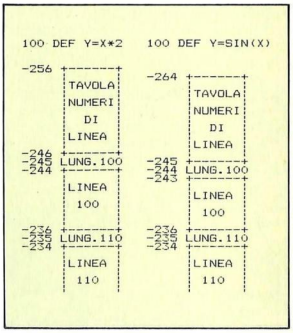


Figura 1 - Il sistema operativo gestisce in modo dinamico gli spazi riservati al programma e alla tavola dei numeri di linea. Editando una linea di programma, non solo cambia il suo indirizzo di partenza, ma viene allocata l'intera Tavola dei numeri di linea.

scrive una linea Def con una funzione che occupi quanto più spazio possibile, in que-

sto modo: Def Y=X+X+X+X..., fino a occupare cinque righe di schermo. Successivamente si accetta in input una stringa contenente una funzione, la si codifica con i valori token caricandola in memoria sopra la precedente e si modifica il byte che contiene la lunghezza della linea interessata. I caratteri eccedenti, cioè X+X+X+..., rimangono per così dire nascosti, non appaiono con un List e vengono del tutto ignorati quando viene calcolato il valore della funzione; essi vengono tuttavia conservati con i comandi Save e Old e restano a disposizione per scrivere formule più o meno lunghe.

Il programma del listato 3 è solo dimostrativo, ma può ben adattarsi ad altri programmi ove sia richiesta una tecnica analoga, a esempio quando si voglia disegnare il grafico di diverse funzioni senza dover interrompere l'esecuzione per ridefinirle. Segue una breve descrizione delle principali routine.

120 - Definizione della funzione.

130 - Dimensionamento delle matrici e caricamento dei valori token.

140-160 - Ricerca dell'indirizzo della linea 120.

170 - Input della funzione.

180-240 - Codifica della funzione con i valori token.

250-260 - Controllo della corrispondenza tra parentesi aperte e chiuse.

270 - Scrittura in memoria dei byte che costituiscono la linea Def.

280-300 - Calcolo della funzione per alcuni valori di X.

310-330 - Data contenenti i valori token.

Il programma accetta solo funzioni nella variabile X ed è protetto contro alcuni errori, ma non possiede un controllo completo della sintassi. Sono errori recuperabili parole riservate diverse da Atn, Cos, Int, Log, Sin, Tan, Pi; errori ortografici, variabili diverse dalla X, parentesi spaiate; sono invece irrecuperabili errori nella struttura della formula. Se si scrive F (X)=X.5+√2 il programma si arresta, compare un messaggio d'errore da parte del sistema e per riprendere bisogna dare nuovamente il Run. ■

Listato 1 - Un piccolo programma d'esempio e il suo output.

```
100 DEF Y=X*2
110 CALL PEEK(-31952,A,B,C,D):: A=A*256+
B-65536 :: C=C*256+D-65536
120 FOR I=C-3 TO A STEP -1 :: CALL PEEK(
I,B,D,E,F):: B=B*256+D :: IF B=100 THEN
140
130 NEXT I :: STOP
140 E=E*256+F-65536 :: I=E-1 :: CALL PEE
K(I,B):: FOR I=E TO E+B-1 :: CALL PEEK(I
,BY):: PRINT BY;:: NEXT I
RUN
 137  89  190  88  195  200  1  50  0
```

Listato 2 - Sostituendo questa riga nel listato precedente si ha un nuovo output.

```
100 DEF Y=SIN(X)

 --------------

RUN
 137  89  190  210  183  88  182  0
```

Listato 3 - Il programma dimostrativo.

```
120 DEF Y=X+X+X+X+X+X+X+X+X+X+X+X+X+X+X+
X+X+X+X+X+X+X+X+X+X+X+X+X+X+X+X+X+X+X+X
X+X+X+X+X+X+X+X+X+X+X+X+X+X+X+X+X+X+X+
X+X+X+X+X+X+X+X+X+X+X
130 CALL INIT :: DIM TV(256),TT$(15,2)::
 FOR J=0 TO 15 :: READ TT$(J,1),TT$(J,2)
:: NEXT J
140 CALL PEEK(-31952,A,B,C,D):: Z=65536
:: A=A*256+B-Z :: C=C*256+D-Z :: I=C-3
150 FOR I=C-3 TO A STEP -4 :: CALL PEEK(
I,B,D,E,F):: B=B*256+D :: IF B=120 THEN
E=E*256+F-Z :: GOTO 170
160 NEXT I :: STOP
170 CALL CLEAR :: INPUT "F(X)=":FN$ :: P
RINT RPT$("-",28):: PRINT :: CTR=0 :: FG
=0
180 FOR J=1 TO LEN(FN$):: W$=SEG$(FN$,J,
1):: Q=ASC(W$):: IF (Q>47 AND Q<58)OR Q=
46 THEN 200 ELSE IF FG=1 THEN TV(CW)=NC
:: FG=0
190 FOR K=0 TO 15 :: P=POS(FN$,TT$(K,1),
J):: IF P<>J THEN 220 ELSE CTR=CTR+1 ::
TV(CTR)=VAL(TT$(K,2)):: J=J+LEN(TT$(K,1)
)-1 :: GOTO 240
200 IF FG=0 THEN CTR=CTR+3 :: TV(CTR-2)=
200 :: CW=CTR-1 :: NC=1 :: TV(CTR)=0 ::
FG=1 :: GOTO 240
210 CTR=CTR+1 :: TV(CTR)=0 :: NC=NC+1 ::
 GOTO 240
220 NEXT K
230 PRINT "ERRORE NELLA FORMULA" :: FOR
TM=1 TO 500 :: NEXT TM :: PRINT :: GOTO
170
240 NEXT J :: IF FG=1 THEN TV(CW)=NC
250 CTR=CTR+1 :: TV(CTR)=0 :: N=CTR :: P
1=0 :: P2=0 :: FOR J=1 TO N :: P1=P1+(TV
(J)=182):: P2=P2+(TV(J)=183)
260 NEXT J :: IF P1<>P2 THEN 230 ELSE I=
E-1 :: BY=N+3 :: CALL LOAD(I,BY)
270 CTR=0 :: FOR I=E+3 TO E+2+N :: CTR=C
TR+1 :: BY=TV(CTR):: CALL LOAD(I,BY):: N
EXT I
280 INPUT "X = ":X :: PRINT "F(X) =";Y :
: PRINT "STESSA FUNZIONE? (Y/N)"
290 CALL KEY(0,K,S):: IF S=0 THEN 290 EL
SE CALL HCHAR(23,3,32,28)
300 IF K=78 THEN 170 ELSE IF K=89 THEN 2
80 ELSE 290
310 DATA ")","182","(","183","=","190","
+","193","-","194","*","195","/","196"
320 DATA "^","197","ATN","204","COS","20
5","INT","207","LOG","208","SIN","210","
TAN","212"
330 DATA "PI","221","X","88"
```

Indice generale e analitico

Anno 1 - n. 1 - Luglio-Agosto 1982

Articoli

- Ventiquattro modi per scrivere un ciclo di W.D. Maurer ... 9
- Master Mind e intelligenza artificiale di D.R. Hope ... 21
- Programmazione ricorsiva in BASIC di A. Leal ... 33
- Conversioni grafiche tra TRS-80, Apple e PET di R. Kaplan ... 53

Programmi
PET,Apple, TRS-80: Gioco del 15 ... 75
PET, TRS-80: Bersaglio ... 79
PET: Orologio digitale ... 81
TRS-80: Colpo strategico ... 83
- File comandi ... 86

Atari: Cuore matto ... 89
- Animazione 3D ... 91
- Caleidoscopio ... 92

Sinclair ZX80: Odissea nello spazio ... 93
- Roulette russa ... 94
- Reverse ... 95
- Rem come dati ... 95

Rubriche

Editoriale
Software vuol dire programmi?
di M. Boscarol ... 7

Giochi informatici
Un piccolo grande problema
di M. Giardino ... 17

Il linguaggio dell'informatica
Da dove vengono i termini dell'informatica ... 41

L'arte di programmare
Proverbi per informatici
di N. Tonolli ... 45

Raccolta di routine BASIC
Punteggiatura ... 51

Libri di software
Una biblioteca per ottimizzare
di G. Staluppi ... 67

Anno 1 - n. 2 - Settembre 1982

Articoli

- Ricerca su alberi
Parte prima - di G. Williams ... 11
- Un Assembler in BASIC per il PET/CBM
di R. Baker ... 33
- L'organizzazione Pert
di W.D. Maurer ... 47
- Immagini digitali di mezzetinte
di J.S. Browning ... 56

Programmi
Apple: Cannonate ... 62
- Ippica ... 66
- Galaxia ... 67

TRS-80: Riflessioni ... 69
- Bombardiere ... 72
- Labirinto mobile ... 74

PET/CBM: Frasi inutili ... 75
TRS-80: Julia ... 77
VIC 20: Star war ... 79
- Crazy car ... 80
- Motocross ... 81

ZX81: Pontoon ... 82
- Derby ... 84
- Musica ... 84

Rubriche

Editoriale
Le mani sulla tastiera
di M. Boscarol ... 7

Il linguaggio dell'informatica
Come tradurre i termini stranieri? ... 31

Giochi informatici
Come farsi umiliare da un computer
di C. Sintini ... 42

Raccolta di routine BASIC
Mescolare ... 46

L'arte di programmare
La programmazione discendente
di N. Tonolli ... 53

Libri di software
Tutti i termini dell'informatica
di G. Staluppi ... 61

Anno 1 - n. 3 - Dicembre 1982

Articoli

- Ricerca su alberi
Parte seconda - di G. Williams ... 26
- La compressione dei testi
di J.L. Peterson ... 36
- Generatore di labirinti
di G. Bond ... 67
- Comunicazione da PET a PET attraverso la User port
di J. Winn ... 73
- Pixelator, generatore di caratteri per il VIC
di J. Calloway ... 78

Programmi
Apple: La carta del cielo
di P. Dennings - versione italiana di M. Boscarol ... 83
TRS-80: Backgammon
di A. Scott - versione italiana di P. Canevarolo ... 89
Apple: Collisione
di A. Giovannetti ... 93
Atom: Rally
di P. Canevarolo ... 96
Apple: Tastiera d'organo
di R. Hausman - versione italiana di P. Canevarolo ... 98
Atari 800: Othello
versione italiana di P. Canevarolo ... 100
- Boing
versione italiana di P. Canevarolo ... 102
- Baseball
versione italiana di P. Canevarolo ... 103
VIC 20: Supercaccia
di P. Canevarolo ... 106
ZX81: Tira e molla
di M. Huber - versione italiana di P. Canevarolo ... 108
- Trappola
di M. Huber - versione italiana di P. Canevarolo ... 109
- Stemma
di M. Huber - versione italiana di P. Canevarolo ... 109
- Pianeta X
di M. Huber - versione italiana di P. Canevarolo ... 110

Rubriche

Editoriale
Le novità di questo mese
di M. Boscarol ... 7

I segreti del personal
Sinclair ZX80: La programmazione dei giochi di movimento
di E. Ferreguti ... 8
TRS-80: Per chi ha il modello 1
di R. Gabrielli ... 10
VIC 20: Doppio schermo sul VIC
di J. Butterfield ... 12
Commodore: Effetti simultanei nei giochi di movimento
di E.M. Albani ... 14

Giochi informatici
La torre di Hanoi
di M. Boscarol ... 20

L'arte di programmare
Regole di stile per autori di software
di N. Tonolli ... 23

Libri di software
Imparare il BASIC dei personal
di G. Staluppi ... 25

Raccolta di routine BASIC
Modifiche e commenti per la routine 1 ... 63

Anno 2 - n. 4 Gennaio-Febbraio 1983

Articoli

BASIC: Interi in precisione multipla
di A. Filz ... 17
- Package di aritmetica intera per il 6502
di M. Cerofolini ... 33

Apple: Grafica tridimensionale con Apple
di M. Pelczarski ... 47

BASIC: Che rischi corre il vostro software?
di J. Commander ... 57

PET/CBM: Gioco del calcio su PET/CBM
di C. Sintini ... 67

Rubriche

Editoriale
Software e idee
di M. Boscarol ... 9

Posta ... 10

Raccolta di routine BASIC
Il controllo del codice fiscale
di M. Boscarol ... 29

Dizionario di BASIC
a cura della Redazione ... 43

I segreti del personal
PET/CBM: L'autoprogrammazione: un primo passo verso l'intelligenza artificiale
di E.M. Albani ... 61
ZX80/ZX81: Tecniche di velocizzazione
di E. Ferreguti ... 64
VIC 20: Come realizzare un listato bidirezionale
di C. Saraceno ... 66

Debug
Il gioco del Nim ... 71
- Il gioco del 15 ... 71

Conversioni
Generatore di labirinti per Apple II e ZX81 ... 73
Piccoli Annunci ... 77

Anno 2 - n. 5 - Marzo-Aprile 1983

Articoli

BASIC: Tecniche di ricerca
di E. Mitchell ... 9
- Le proposte per il nuovo BASIC Standard
di M. Boscarol ... 19

Atari: Comporre musica con l'Atari
di W.L. Colsher ...

Apple: Il sistema Pretty Printer per l'Apple
di S. Ventura ... 27

Generico: Duplicazione abusiva, pericolo reale o timore infondato?
di J. Grout ... 35

ZX Spectrum: Contraerea: le capacità grafiche dello ZX Spectrum
di M. Spero ... 40

ZX81: Minicalc per lo ZX81
di E. Ferreguti ... 50

Apple: Grafici sempre a portata di mano
di S. Brown ... 58

Rubriche

Editoriale
Il software di oggi e di domani
di M. Boscarol ... 5

Posta ... 6

Raccolta di routine BASIC
Il controllo del numero di Partita IVA
di M. Boscarol ... 17

Dizionario di BASIC
a cura della Redazione ... 31

I segreti del personal
PET/CBM: Tempus fugit...
di E.M. Albani ... 43
ZX80/ZX81: Read, Data e Restore e l'uso del registratore
di E. Ferreguti ... 45
ZX Spectrum: Le più interessanti caratteristiche di gestione video
di M. Spero ... 46
VIC 20: Disabilitazione del tasto Run/Stop
di A. Guida ... 49

Conversioni
PET/CBM: La carta del cielo
di N. Faccini ... 54
Piccoli Annunci ... 63

Anno 2 - n. 6 - Maggio 1983

Articoli

BASIC: Un piccolo parallelo tra pocket BASIC
di P. Capobussi ... 8
- Assolutizzare le etichette
di P. Capobussi ... 16
- Operazioni, precisioni, approssimazioni
di P. Capobussi ... 19

PC 1211/TRS pocket: Istogramma universale per PC 1211/TRS80 pocket
a cura della Redazione ... 21
TI58/59: Calcoli ingegneristici con TI58/59
di A. Scifoni ... 22
BASIC: Brevi ricette per ottimizzare
di P. Capobussi ... 28
TI58/59: Quadratini e lacune per TI58/59
di F. Carbone

Indice generale e analitico

HP: Tre programmi di chimica e fisica con l'HP67 - HP97
di F. Sismondo ... 33
BASIC: La compressione dei dati
di P. Capobussi ... 39
HP: Master Mind su HP-29C
di E. Bossaglia ... 42
TRS-80 pocket: Interfacciamento: TRS-80 pocket contro TRS-80 Mod. 1
a cura della Redazione ... 47
TI59: Moduli e fasi di funzioni di trasferimento con TI59
di G. Fedecostante ... 51
Casio: Logica aristotelica con la calcolatrice tascabile
di G. De Nicolao ... 55

Rubriche

Editoriale
Il fascino dei computer
di M. Boscarol ... 7
Conversioni
Giochiamo d'azzardo sulle PC-1500
di E. Cima ... 62
Piccoli Annunci ... 65

Anno 2 - n. 7 - Giugno 1983

Articoli

ZX81: Le variabili di sistema nello ZX81
di B. Del Medico ... 15
● Project robot: il robotwar per lo ZX81
di E. Ferreguti ... 21
ZX Spectrum: Guida alla conversione dallo ZX81 allo ZX Spectrum
di M. Spero ... 27
TRS-80: La trasformazione automatica di coordinate topografiche
di B. Saccà ... 35
VIC 20: Come stampare lo schermo del VIC
di A. Guida ... 39
Apple: Un data base modulare per l'Apple
di M. Pelczarski ... 47
Generico: Dal BASIC al Pascal
Parte prima - di R. W. Anderson ... 63

Rubriche

Editoriale
Alle frontiere della scienza dei computer
di M. Boscarol ... 7
Posta ... 10
Raccolta di routine BASIC
Giorno della settimana e fase della luna
di M. Boscarol ... 33
Recensioni
Avventura nel castello: un'adventure ... 45
I segreti del personal
ZX Spectrum : L'uso della tastiera nei programmi di movimento
di M. Spero ... 53
TI99/4A: Input e stampa estesa
di F. Cerulo ... 55
VIC 20: Routine di interrogazione dei joystick per il VIC 20
di F. Cordes ... 56
PET/CBM: Overlay
di E.M. Albani ... 57
Nuova Elettronica: Esa e ASCII
di V. Scaffidi ... 60
Apple: Applesoft Cataloger
di R. Amasino ... 61
ZX81: Caratteri minuscoli su ZX Printer
di E. Ferreguti ... 62
Conversioni
VIC 20: Collisione per il VIC
di A. Casale ... 68
● Torre di Hanoi animata per il VIC
di M. Quintini ... 71
Piccoli Annunci ... 73

Anno 2 - n. 8/9 - Luglio-Agosto 1983

Articoli

ZX80/ZX81: Conversione di programmi per ZX80 e ZX81
Parte prima - di B. Del Medico ... 9
VIC 20: Master Mind contro VIC
di P. Pulli ... 17
TI99/4A: Programmare grafici con il TI99/4A
di R. Del Giudice ... 23
VIC 20: Picture
di M. Minischetti ... 25
C 64: Quattro programmi per C 64
di C. Sintini ... 27
PET/CBM: Gestire file con il PET/CBM
di S. De Monte ... 39
TI99/4A: Archivio scacchistico per TI99/4A
di S. Borsani ... 43
VIC 20: Procedura di recupero
di L. Gemme ... 49
ZX81: Guida e cacciatore
di P. Ferrari ... 51
● ZX Manager
di E. Ferreguti ... 54
● ZX data base
di E. Ferreguti ... 58
● Parliamo
di F. Manfredini ... 62
DAI: Il giro del cavallo
di G. Morpurgo ... 64
Generico: Dal BASIC al Pascal
Parte seconda - di R.W. Anderson ... 72
TRS-80: Matematica aerobica con il TRS-80
di B. Douglas ... 77

Rubriche

Editoriale
Come passerò l'estate
di M. Boscarol ... 5
Posta ... 6
Raccolta di routine BASIC
I nidi For
di M. Boscarol ... 83
I segreti del personal
PET/CBM: Un tasto per un'istruzione
di G. Franza ... 85
TI99/4A: Print e Input in un contesto grafico
di S. Borsani ... 88
ZX81: Scritte giganti, grafici e disegni, routine e messaggi video
di E. Ferreguti ... 90
VIC 20: Gestione del cursore del VIC 20
di E. Bossaglia ... 93
ZX Spectrum: Come programmare uno schermo scorrevole
di M. Spero ... 96
Conversioni
Julia ... 101
Debug
Come realizzare un listato bidirezionale ... 103
Contributi dei lettori
Il controllo del Codice fiscale ... 105
● Utilizzare le routine della cartridge del VIC ... 108
Piccoli Annunci ... 110

Anno 2 - n. 10-11 Settembre-Ottobre 1983

Articoli

ZX80/ZX81: Conversione di programmi per ZX81 e ZX80 nuova ROM
Parte seconda - di B. Del Medico ... 14
Generico: Introduzione all'intelligenza artificiale
Parte prima - di B. Del Medico ... 22
Generico: I programmi di simulazione
di F. Sardo ... 40
ZX81: Giochi di inseguimento
di M. Morchio ... 44
● Area di linguaggio macchina nel BASIC ZX81
di M. Petraccone ... 46
BASIC: Accesso per chiave logica a un archivio relative
di G. Confortini ... 48
VIC 20: Dedalo 3D
di A. Guida ... 54
TI99/4A: Il taglio del TI99/4A
di A. Chessa ... 56

Rubriche

Editoriale
Il grimaldello elettronico
di R. Paolillo ... 7
Posta ... 10
I segreti del personal
VIC 20: Restore nn per il VIC 20
di A. Guida ... 62
PET/CBM: Un overlay più flessibile
di A. Zannoni ... 64
ZX81/ZX Spectrum: Risparmiamo memoria
di B. Del Medico ... 64
Conversioni
TI99/4A: Musica con TI99/4A ... 70
ZX Spectrum: Labirinto per Spectrum ... 71
ZX81: Rally per ZX81 ... 71
VIC 20: Project-Robot ... 71
Contributi dei lettori
Rotazione bidirezionale sul video dello ZX81 ... 74
● Operatori logici per TI99/4A ... 78
Debug
Guida e cacciatore ... 79
● Sistemi ridotti per il Totocalcio per C 64 ... 79
Piccoli Annunci ... 80

Anno 2 - n. 12-13 Novembre-Dicembre 1983

Articoli

ZX80/ZX81: Conversione di programmi per ZX81 e ZX80 nuova ROM
Parte terza - di B. Del Medico ... 8
C 64: Labirinto di ghiaccio e Dot
di F. Stella ... 14
ZX Spectrum: Tecniche di input e output per lo ZX Spectrum
di M. Spero ... 20
C 64: Programmer's tool-kit per Commodore 64
Parte prima - di A. Guida ... 24
ZX Spectrum: Asteroidi per ZX Spectrum
di M. Spero ... 30
PET/CBM: "Wei-ch'i", un gioco cinese per PET
di U. Barzaghi ... 34
ZX81: Linguaggio macchina per ZX81
Parte prima - di B. Del Medico ... 52
● ZX81 Superscientifico
di D. Di Mario ... 64
TI99/4A: I file su cassetta nel TI99/4A
di F. Cerulo ... 66
ZX Spectrum: Come salvare capra e cavoli nello ZX Spectrum
di F. Sardo ... 74

Rubriche

Editoriale
Gente comune
di R. Paolillo ... 5
Posta ... 7
I segreti del personal
Atom-Acorn: Controllo della posizione del cursore
di I. Albanese ... 78
VIC 20/C 64: Istruzione On Error Goto nn per il VIC 20 e C 64
di A. Guida ... 80
TI99/4A: Rette ad alta risoluzione
di S. Borsani ... 82
● Videografici
di S. Borsani ... 84
ZX Spectrum: La funzione Screen$: come utilizzarla anche per caratteri grafici
di M. Spero ... 88
Contributi dei lettori
ZX Spectrum: Effetti sonori particolari
di M. Carletti ... 91
TI99/4A: Come controllare la memoria del TI99/4A
di P. Roncali ... 92
Piccoli Annunci ... 95

Anno 3 - n. 14 - Gennaio 1984

Articoli

VIC 20: VIC Pilot
di F. Stella ... 8
Apple: Probabilità e frequenza relativa
di R. Mazzurco e M. Flora ... 14
C 64: Programmer's tool-kit per Commodore 64
Parte seconda - di A. Guida ... 18
Apple: File analyzer
di S. Orlando ... 27
TI58-59: Semplificazione di frazioni per TI58-59
di F. Carbone ... 37

Indice generale e analitico

Generico: Memorie associative e ordinamento
di M. Coccorese 41
TI99/4A: Grafici ad alta risoluzione con il TI99/4A
di S. Borsani 47
ZX Spectrum: Othello per ZX Spectrum
di S. Cerutti 53
ZX81: Dama cinese e Motocross
di C. Driussi 56
• Piani di ammortamento mutui con lo ZX81
di A. Motta 59
ZX Spectrum: Un esempio di analisi lessicografica
di F. Sardo 63
• Alcuni trucchi dello Spectrum
di T. Policastro 65

Rubriche

Editoriale
L'ora di informatica
di R. Paolillo 5
Posta 7
I segreti del personal
VIC 20/C 64: Disabilitazione List: la nuova funzione di Append
di I. Albanese 68
• Come realizzare la funzione Append
di L. Iacono 69
ZX Spectrum: Come proteggere i vostri programmi
di M. Spero 70
Sharp: Tre nuove istruzioni
di M. Lenzi 72
TI99/4A: Il movimento del TI BASIC
di S. Borsani 73
Debug 77
Piccoli Annunci 79

Anno 3 - n. 15 - Febbraio 1984

Articoli

ZX81: Linguaggio macchina per Sinclair
Parte seconda - di B. Del Medico 8
PET/CBM: Abukir 1798, battaglia navale per CBM
Parte prima - di U. Barzaghi 20
Generico: Dal BASIC al Pascal
Parte terza - a cura della Redazione 28
VIC 20/C 64: Impariamo il linguaggio macchina con il VIC 20 e il C 64
Parte prima - di A. Guida 30
Generico: Introduzione all'intelligenza artificiale
Parte seconda - di B. Del Medico 36
ZX Spectrum: Superman
di I. Parbuono 54
DAI: Un potente word processor
di G. Morpurgo 56
Apple: Ziup, zip e swoop
di A. Stecchina 67
ZX Spectrum: Semplice renumber per Spectrum a cura della Redazione 70
• Ingrandimento e riduzione di caratteri
di M. Spero 72

Rubriche

Editoriale
Alla ricerca del regalo intelligente
di R. Paolillo 5
Posta 6
I segreti del personal
Sharp: La mappa di memoria del PC-1251
di M. Lenzi 78
ZX Spectrum: Come incolonnare correttamente i numeri senza ricorrere a una subroutine
di M. Spero 79
TI99/4A: Il movimento in TI Extended BASIC
di S. Borsani 81
VIC 20/C 64: La gestione della tastiera
di A. Guida 85
Conversioni
ZX Spectrum: Collisioni per ZX Spectrum
di M. Morchio 91
Piccoli Annunci 95

Anno 3 - n. 16 - Marzo 1984

Articoli

ZX81: Routine in linguaggio macchina per ZX81
Parte prima - di C. Cappelli 8
ZX Spectrum: Ideazione di caratteri speciali
di B. Del Medico 18
C 64: Music Editor
Parte prima - di M. Gremes 24
ZX81: Frogger per Sinclair ZX81
di G. Tisi 31
• Dama cinese per Sinclair ZX81
di A. Motta 34
• Arriva una nave carica di... programmi da 1 Kbyte per ZX81
di B. Del Medico 37
• Giochi africani per Spectrum e ZX81
di B. Del Medico 45
Generico: Dal BASIC al Pascal: funzioni di ingresso/uscita
Parte quarta - a cura della Redazione 62
Spectrum: Calcoli relativi alla distribuzione normale
di T. Policastro 67
VIC 20: I boss del petrolio
Parte prima - di E. Comini 70
C 64: Sprite per C 64
Parte prima - di F. Stella 75
PET/CBM: Abukir 1798
Parte seconda - di U. Barzaghi 82

Rubriche

Editoriale
Una lingua per tutti
di R. Paolillo 5
Posta 7
I segreti del personal
VIC 20/C 64: La gestione dei dati sul VIC e C 64
di A. Guida 105
ZX Spectrum: Risparmio di memoria con insiemi di numeri interi
di M. Spero 108
Sharp: Alla scoperta di nuovi caratteri
di M. Lenzi 110
Piccoli Annunci 112

Anno 3 - n. 17 - Aprile 1984

Articoli

ZX Spectrum: Rally Safari
di I. Parbuono 8
C 64: Music Editor
Parte seconda - di M. Gremes 14
Generico: Dal BASIC al Pascal
Parte quinta - a cura della Redazione 20
Apple: Gestione del video tramite Peek e Poke
di C. Poma 24
VIC 20/C 64: Impariamo il linguaggio macchina con il VIC 20 e il C 64
Parte seconda - di A. Guida 26
ZX81: Ordinamenti con lo ZX81
di A. De Santis 32
PET/CBM: La battaglia del lago ghiacciato
Parte prima - di U. Barzaghi 34
ZX81: Othello per ZX81
Parte prima - di A. Motta 44
• Satel per ZX81
di A. De Santis 54
C 64: Sprite per C 64
Parte seconda - di F. Stella 56
ZX Spectrum: Spacman per ZX Spectrum
di I. Parbuono 63
Apple: Macro istruzioni per Apple II
di R. Brunialti 66
VIC 20: Il VIC ben temperato
di F. Pozzi 76
• Missione Ufo
di R. Comini 80

Rubriche

Editoriale
Uno sguardo dietro l'angolo
di R. Paolillo 5
Posta 7
I segreti del personal
Sharp: Alla ricerca del comando nascosto
di M. Lenzi 84
TI99/4A: Print Using in TI BASIC
di S. Borsani 85
ZX Spectrum: Eliminare blocchi di programmi senza cancellare le linee singolarmente
di M. Spero 87
C64: Gli "orologi" nel C 64
di A. Guida 89
Piccoli Annunci 9

Anno 3 - n. 18 - Maggio 1984

Articoli

PET/CBM: La battaglia del lago ghiacciato
Parte seconda - di U. Barzaghi 8
ZX81: ZX81 Croupier
di A. Motta 24
VIC 20: Type-Writer 1
Parte prima - di A. Guida 30
Apple/C 64: Schermo Apple-C 64 e la ragnatela
di C. Poma 37
C 64: Il C 64 in italiano
di A. Guida 43
ZX Spectrum: Analisi delle serie temporali
di V. Riva 51
VIC 20/C 64: Impariamo il linguaggio macchina con il VIC 20 e il C 64
Parte terza - di A. Guida 57
ZX81: Othello per ZX81
Parte seconda - di A. Motta 63
ZX Spectrum: Robot a caccia
di M. Morchio 69
TI99/4A: Grafica avanzata sul TI 99/4A
di F. Cerulo 75
VIC 20/C 64: Othello-Reversi per VIC 20 e C 64
di A. Guida 81
ZX81: Routine in linguaggio macchina per ZX81
Parte seconda - Di C. Cappelli 87

Rubriche

Editoriale
Le regole del gioco
di R. Paolillo 5
Posta 6
I segreti del personal
TI99/4A: Diagrammi in TI BASIC
di S. Borsani 100
C 64: Il C 64 e la memoria nascosta
di A. Guida 101
Sharp: Giochiamo con i numeri di linea
di M. Lenzi 104
ZX Spectrum: List e Copy in doppia altezza
di M. Spero 105
Contributi dei lettori
ZX81: Mini monitor per ZX81
di M. Cappelletti 109
ZX Spectrum: Conversione veloce decimale-binario
di M. Spada 109
Piccoli Annunci 111

Anno 3 - n. 19 - Giugno-Luglio 1984

Articoli

ZX Spectrum: Controllo del video dal linguaggio macchina
di W. Sini 12
• Lista delle variabili per Spectrum
di R. Galliena 14
• Loquax per Spectrum
di C. Giavani 16
• Nimble per Spectrum
di F. Ghedini 20
VIC 20: Tipe-Writer 1
Parte seconda - di A. Guida 30
C 64: Death per C 64
di L. Marras 40
DAI: Indagine su alcune strane figure
di G. Morpurgo 46
ZX81: Calcolo letterale di polinomi
di C. Cappelli 65
C 64: Rivisitiamo insieme Programmer's tool kit
di A. Guida 70
Generico: Dal BASIC al Pascal
Parte sesta - a cura della Redazione 76
TI99/4A: Sprite Editor per TI99/4A
di F. Cerulo 80
• Uso delle tabelle con TI99/4A
di S. Borsani 86
VIC 20/C 64: Impariamo il linguaggio macchina con il VIC 20 e il C 64
Parte quarta - di A. Guida 92

Rubriche

Editoriale
Le domande imbarazzanti
di R. Paolillo 5

Indice generale e analitico

Posta ... 6
Personal news
a cura di M. Giacobazzi ... 8
I segreti del personal
Sharp: I numeri misteriosi
di M. Lenzi ... 98
TI99/4A: Gestione file con un solo registratore
di S. Borsani ... 99
ZX Spectrum: Semplificazione della procedura di In da tastiera
di M. Spero ... 102
VIC 20/C 64: Magic Key
di A. Guida ... 105
Piccoli Annunci ... 109

Anno 3 - n. 20 Agosto-Settembre 1984

Articoli

CBM: Rally
Parte prima - di U. Barzaghi ... 12
ZX Spectrum: Studiare con lo Spectrum
di M. Morchio ... 17
Generico: Dal BASIC al Pascal
Parte settima - a cura della Redazione ... 20
TI99/4A: Stid Poker
di G. Persiano ... 28
PET/CBM/C 64: Gioco della Scopa a 3 carte
Parte prima - di S. Guarinelli ... 37
C 64: Salti etichettati per C 64
di M. Paolinelli ... 48
Sharp: Contabilità personale
di G. Gatti ... 52
Generico: Intelligenza artificiale e creatività
di C. Poma ... 57
ZX Spectrum: Walter il raccoglimele
di I. Parbuono ... 65
VIC 20/C 64/CBM: Stampiamo le ricevute fiscali
di S. De Monte ... 72
C 64: Città per C 64
di S. Borsani ... 85
VIC 20/C 64: Impariamo il linguaggio macchina con il VIC 20 e il C 64
Parte quinta - di A. Guida ... 93
ZX81: Le torri di Plutone
di A. Motta ... 99

Rubriche

Editoriale
Chi trova un amico
di R. Paolillo ... 5
Posta ... 6
Personal news
a cura di M. Giacobazzi ... 10
I segreti del personal
VIC 20/C 64: Procedura autostart
di A. Guida ... 108
Sharp: Le subroutine del sistema operativo
di M. Lenzi ... 112
ZX Spectrum: On Error Goto e disabilitazione del tasto Break
di M. Spero ... 114
Contibuti dei lettori ... 118
Conversioni ... 121
Piccoli Annunci ... 124

Anno 3 - n. 21 - Ottobre 1984

Articoli

ZX Spectrum: Pluri-Calc
di S. Cerutti ... 12
● Simulazione di joystick da tastiera
di M. Modelli ... 20
● Risoluzione triangoli con lo Spectrum
di A. Furlan ... 32
C 64/VIC 20: Da un Commodore all'altro
di U. Barzaghi ... 39
ZX Spectrum: Cariche
di F. Andreotta e C. Testa ... 44
PET/CBM: Rally
Parte seconda - di U. Barzaghi ... 48
Sharp: Schedina 13 per Sharp MZ-700
di G. Gatti ... 75
PET/CBM/C 64: Scopa a tre carte
Parte seconda - di S. Guarinelli ... 82
VIC 20/C 64: Impariamo il linguaggio macchina con il VIC 20 e il C 64
Parte sesta - di A. Guida ... 111
Generico: Guarda un po' come si insegna la musica al personal ... 115

Rubriche

Editoriale
Standard o non standard?
di R. Paolillo ... 5
Posta ... 6
Personal news
a cura di M. Giacobazzi ... 8
I segreti del personal
VIC 20/C 64: Una raffica di trucchi
di A. Guida ... 116
Sharp: I byte di controllo del display
di M. Lenzi ... 118
ZX Spectrum: Routine in linguaggio macchina per gestione video
di M. Spero ... 120
TI99/4A: Apertura di un file nel modo Append
di S. Borsani ... 121
Piccoli Annunci ... 123

Anno 3 - n. 22 - Novembre 1984

Articoli

Apple: Giochiamo a Bridge con Apple II
di V. Delle Cave ... 15
BASIC: Atterraggio
di M. Sangiorgio ... 22
Apple: Calcolo combinatorio in multipla precisione
di R. Mazzurco ... 44
ZX Spectrum: ZX Expert
Parte prima - di L. Lotti ... 58
C 64: Awele
di L. Baumer ... 62
ZX Spectrum: Routine per Spectrum
di I. Parbuono ... 68
Generico: Expandet Sound Editor: che musica! ... 73
VIC 20/C 64: Impariamo il linguaggio macchina con il VIC e il C 64
Parte settima - di A. Guida ... 75
TI99/4A: Ripartizione spese di riscaldamento
di S. Borsani ... 79
C 64: Il casello di Melegnano per C 64
di F. Sardo ... 91

Rubriche

Editoriale
Quando il personal dà spettacolo
di R. Paolillo ... 5
Posta ... 7
Personal news
a cura di M. Giacobazzi ... 10
I segreti del personal
VIC 20/C 64: Alcune scoperte sul drive 1541
di A. Guida ... 97
Sharp: La rappresentazione dei numeri
di M. Lenzi ... 110
TI99/4A: Accesso diretto in un file sequenziale ?!
di S. Borsani ... 115
ZX Spectrum: End of file: da errore a funzione
di M. Spero ... 118
Piccoli Annunci ... 122

Anno 3 - n. 23 - Dicembre 1984

Articoli

Sharp: Tombola per Sharp MZ-700
di M. Sangiorgio ... 14
C 64: Space Traveller
di G. Peritore ... 18
● Dentro l'avventura
Parte prima - di R. Tabacco ... 35
ZX Spectrum: Memorizzazione di immagini
di S. Pavanello ... 40
C 64: Frogger per C 64
di L. Marras ... 44
ZX Spectrum: ZX Expert
Parte seconda - di L. Lotti ... 50
Apple: Torneo di Bridge
Parte prima - di V. Delle Cave ... 57
TI99/4A: Atterraggio pericoloso per TI99
di M. Cristuib Grizzi ... 64
Generico: Concerto grosso per C 64 e sintetizzatori con Multitrack Composer ... 69

Rubriche

Editoriale
Il gusto dell'avventura
di R. Paolillo ... 5
Posta ... 6
Personal news
a cura di M. Giacobazzi ... 8
I segreti del personal
VIC 20/C 64: I super BASIC
Parte prima - di A. Guida ... 70
TI99/4A: Sottoprogrammi in TI Extended BASIC
di S. Borsani ... 80
ZX Spectrum: Trucchi e bug per tutti i gusti
di M. Spero ... 84
Piccoli Annunci ... 90

Anno 4 - n. 24 - Gennaio 1985

Articoli

ZX Spectrum: Titolazione con lo Spectrum
di P. Coretti ... 10
TI99/4A: Word processor per TI99/4A
di S. di Tommaso ... 21
ZX Spectrum: Trasformate il vostro Spectrum in un registratore digitale ... 35
Sharp: Testi personalizzati per Sharp MZ-700
di G. Gatti ... 36
C 64: Alta risoluzione
di A. Bartalesi ... 46
Apple: Grafica e joystick per Apple II
di C. Poma ... 52
● Torneo di Bridge
Parte seconda - di V. Delle Cave ... 60
ZX Spectrum: Archivio per Spectrum
di N. Carloni ... 66
C 64: Come aggiungere nuovi comandi BASIC al C 64
di M. Paolinelli ... 73
VIC 20/C 64: Impariamo il linguaggio macchina con il VIC 20 e il C 64
Parte ottava - di A. Guida ... 82
C 64: Dentro l'avventura
Parte seconda - di R. Tabacco ... 92

Rubriche

Editoriale
La città informatica
di R. Paolillo ... 5
Posta ... 6
Personal news
a cura di M. Giacobazzi ... 8
I segreti del personal
VIC 20/C 64: I super BASIC
Parte seconda - di A. Guida ... 96
Sharp: Alla scoperta del linguaggio macchina
di M. Lenzi ... 100
ZX Spectrum: L'uso e... l'abuso dei caratteri di controllo
di M. Spero ... 102
TI99/4A: Ring
di S. Borsani ... 105
Piccoli Annunci ... 112

Anno 4 - n. 25 - Febbraio 1985

Articoli

ZX Spectrum: Language Tutor per Spectrum 48 Kbyte
di M. Mannuzzi ... 14
C 64: Sprite Editor
di C. Tovena ... 22
Sharp: Organo elettronico
di M. Sangiorgio ... 40
ZX81: Dama per ZX81
di A. Motta ... 46
ZX Spectrum: Il castello
di L. Lotti ... 50

Indice generale e analitico

TI99/4A: Roulette
di M. Cristuib Grizzi 58
VIC 20: One Touch per VIC 20
di G. Bellegatti 62
C 64: Super Assembler
di G. Puccio 66
VIC 20: Phantoms labirinto
di E. Bove 80
ZX Spectrum: Super Converter per Spectrum
di S. Cerutti 84
Apple: Apple: pagine video e dintorni
di G. Pisani 88

Rubriche

Editoriale
Un chilo di software
di R. Paolillo 5
Posta 7
Personal news
a cura di M. Giacobazzi 10
I segreti del personal
VIC 20/C 64: Type-Writer e dimensionamento dinamico
di A. Guida 102
TI99/4A: Recupero degli errori
di S. Borsani 103
ZX Spectrum: Extended BASIC
di M. Spero 107
Piccoli Annunci 113

Anno 4 - n. 26 - Marzo 1985

Articoli

Sharp: Catalogo nastri
di M. Sangiorgio 12
Apple: Gestione archivio fotografico con Apple
di R. Cobianchi 20
C 64: Dentro l'avventura
Parte terza - di R. Tabacco 34
ZX81: Aliens per ZX81
di A. Motta 40
ZX Spectrum: TTmusic
di TTsoft 44
C 64: L'ultimo perde
di S. Borsani 48
ZX Spectrum: Memo-Test per Spectrum
di P. Finizio 52
TI99/4A: Bioritmi
di D. Sgorbini 54
C 64: Metodi di Newton
di A. Bosoni 60
Apple: Sistemi ridotti Totocalcio
Parte prima - di V. Delle Cave 64
ZX Spectrum: Quadrati per Spectrum
di A. Furlan 72

Rubriche

Editoriale
Per qualche contatto in più
di R. Paolillo 5
Posta 7
Personal news
a cura di M. Giacobazzi 8
I segreti del personal
VIC 20/C 64: Beep e errori "sonorizzati"
di M. Paolinelli 86
Sharp: Come sproteggere i programmi
di M. Lenzi 88
TI99/4A: Sub copy
di S. Borsani 89
Piccoli Annunci 92

Anno 4 - n. 27 - Aprile 1985

Articoli

● **TI99/4A:** Text-Editor per TI99/4A
di S. Borsani 12
C 64: Equadif per C 64
di S. Checchini 25
● Gollum
Parte prima - di S. Guarinelli 37
ZX Spectrum: Baskit per Spectrum
di S. Cerutti 43
Apple: Sistemi ridotti Totocalcio
Parte seconda - di V. Delle Cave 59
ZX Spectrum: Oroscopo per Spectrum
di I. Parbuono 65
C 16: La battaglia del lago ghiacciato per C 16
di A. Motta 69
C 64: Dentro l'avventura
Parte quarta - di R. Tabacco 75
ZX Spectrum: Graphkit per Spectrum
di R. Pesce 79
Sharp: Piani d'ammortamento con lo Sharp
di M. Sangiorgio 85

Rubriche

Editoriale
Il tempo perduto
di R. Paolillo 5
Posta 7
Personal news
a cura di M. Giacobazzi 9
I segreti del personal
Spectrum: Diamo ai programmi BASIC la massima qualità nella grafica di animazione
di M. Spero 97
VIC 20/C 64: Gli interrupt del chip video del C 64
di A. Guida 102
Sharp: Autostart e copia programmi
di M. Sangiorgio 106
Piccoli Annunci 110

Anno 4 - n. 28 - Maggio 1985

Articoli

ZX81: Routine in linguaggio macchina
di C. Cappelli 12
TI99/4A: Caccia al tredici
di C. Aceti 37
Apple/C 64/C 16: Caccia alla coppia per Apple II, C 64, C 16
di C. Poma 41
Sharp: Grafici di funzioni trigonometriche
di F. Farina e D. Zamparioio 50
Apple: Bianchi & Neri per Apple
di S. Guarinelli 55
C 64: CMK 49: note e musica con il Commodore 64 62
ZX Spectrum: Grendel, il vostro renumber
di S. Cerutti 63
C 64: Gollum
Parte seconda - di S. Guarinelli 70
● Skywalker per C 64
di L. Marras 87
ZX Spectrum: Zero e croci con lo Spectrum
di A. Furlan 93

Rubriche

Editoriale
Il nostro medioevo
di R. Paolillo 5
Posta 7
Personal news
a cura di M. Giacobazzi 9
I segreti del personal
VIC 20/C 64: Appunti di utilità e Gosub con tasti funzione
di A. Guida 101
ZX Spectrum: Come (tentare di) recuperare programmi BASIC
di M. Spero 104
Sharp: Struttura interna delle istruzioni BASIC
di M. Sangiorgio 108
Piccoli Annunci 112

Anno 4 - n. 29 - Giugno 1985

Articoli

C 64: RAM test
di P. Agostini 12
C 16: Super spreadsheet
di A. Motta 17
TI99/4A: I Ching: il Libro dei Mutamenti
di D. Cerù 28
ZX Spectrum: Progetto aeromodello "U-Control"
di P. Ferrami 34
MSX: Musica con un MSX
di P. Ferrami 49
ZX Spectrum: Tasti funzione per Spectrum 48 Kbyte
di M. Modelli 53
Apple: Nuovo torneo di Bridge
di V. Delle Cave 58
C 64: Monitor 64
di G. Puccio 68
ZX Spectrum: Quoziente intelligenza
di I. Parbuono 96

Rubriche

Editoriale
Oltre il personal
di R. Paolillo 5
Posta 6
Personal news
a cura di M. Giacobazzi 8
I segreti del personal
TI99/4A: Palla che rimbalza
di S. Borsani 102
C 64: File Examiner
di P. Agostini 104
Sharp: L'architettura del sistema
di M. Lenzi 109
Personal Market 112

Anno 4 - n. 30 - Luglio-Agosto 1985

Articoli

C 64: La simulazione dinamica dei fenomeni continui
Parte prima - di F. Sardo 16
Casio: Agenda telefonica
di F. Fanecco 24
C 16: Data base
di C. Poma 27
C 64: Easy Video 64
di M. Paolinelli 41
ZX Spectrum: Cubo magico
di Compagiani 47
TI99/4A: Hi-Res
di S. Borsani 52
Apple: Statistica a più dimensioni
di R. Brunialti 61
ZX Spectrum: Geo-Race
di F. Lo Cicero 77
● Routine 2 per Spectrum
di I. Parbuono 81
MSX: Disegnatore per MSX
di P. Ferrami 86

Rubriche

Editoriale
Musica, maestro!
di R. Paolillo 7
Posta 9
Personal news
a cura di M. Giacobazzi 12
I segreti del personal
C 64: Uso delle label simboliche
di L. Squarza 96
Sharp: Caratteri & colori
di M. Sangiorgio 102
ZX Spectrum: Da cassetta a microdrive
di M. Spero 104
TI99/4A: Quali segreti ci riserva la stampa?
di S. Borsani 107
Personal Market 110

Anno 4 - n. 31 - Settembre 1985

Articoli

C 16: Box
di A. Motta 14
Apple: Scopa a tre carte
di E. Chiave 22
Sharp: Calcolo delle resistenze
di F. Farina e D. Zamparioio 39
ZX Spectrum: Sprite
di M. Modelli 44
C 64: Grand Prix
Parte prima - di U. Barzaghi 55

Indice generale e analitico

MSX: Progetto aeromodello per volo vincolato
di P. Ferrami 75
C 64: DIEL - Data base sinthesizer 95

Rubriche

Editoriale
La banca telematica
di R. Paolillo 5
Posta 6
Personal news
a cura di M. Giacobazzi 8
I segreti del personal
ZX Spectrum: Screen Input
di M. Spero 86
C 64: On interval Gosub C 64
di A. Caporin 90
Sharp: Programmiamo in Assembler
di M. Lenzi 93
Personal Market 96

Anno 4 - n. 32 - Ottobre 1985

Articoli

ZX Spectrum: Animazioni di funzioni tridimensionali
di C. Cappelli 12
Atari 800XL: Destroy the city con l'Atari
di D. Cerù 18
ZX Spectrum: La gestione del video
di C. Cappelli 20
MSX: Uno sguardo nella VDP RAM
di S. Borsani 45

C 64: Prospettiva e grafica tridimensionale
di S. Checchini 52
Sharp MZ-700: Intestazione cassette
di P. Rossi 63
Apple: Elaborazione a quattro mani
di P. Morocutti 66
C 64: La simulazione dinamica di fenomeni continui
Parte seconda - di F. Sardo 73

Rubriche

Editoriale
Crimini informatici
di R. Paolillo 5
Posta 6
Personal news
a cura di M. Giacobazzi 8
Personal Market 80

Anno 4 - n. 33 - Novembre 1985

Articoli

C 64: Music Editor
di T. Palermo 14
Apple, ZX Spectrum: Cartelle per tutti
di D. Cerù 26
ZX Spectrum: La computer art secondo lo Spectrum
di C. Cappelli 30
MSX, M20: Un buco nell'acqua
di S. Borsani 40
TI99/4A: Bogey Boogie
di P. Chianura 44
VIC 20: Totocalcio: sviluppo colonnare condizionato
di V. Carnemolla 52

Rubriche

Editoriale
Noi e gli altri
di R. Paolillo 5
Posta 8
Personal News
a cura di M. Giacobazzi 11
Le Aziende Informano 58
Personal Market 60

Anno 4 - n. 33 - Dicembre 1985

Articoli

Sharp MZ-700: Progettazione di filtri attivi
di D. Zamparíolo 11
ZX Spectrum: Sprite per disegnare
di M. Modelli 22
VIC 20: Far West
di M. Pasquale 32
C 64: Grand Prix
Parte seconda - di U. Barzaghi 37
Apple: Battaglia galattica
di M. Sessa 53

Rubriche

Editoriale
La rivincita del personal
di R. Paolillo 5
Personal News
a cura di M. Giacobazzi 8
Personal Market 65

Indice generale software

Utility

Titolo	P.S n.	Pag.
Apple		
Il sistema Pretty Printer per l'Apple	5	27
Applesoft Cataloger	7	61
File analyzer per Apple	14	27
Ziup, zip e swoop	15	67
Gestione del video tramite Peek e Poke	17	24
Macro istruzioni per Apple II	17	66
Torneo di Bridge 1	23	57
Apple: pagine video e dintorni	25	88
BASIC		
Punteggiatura	1	51
Ventiquattro modi per scrivere un ciclo	1	9
Conversioni grafiche tra TRS-80, Apple e PET	1	53
Mescolare	2	46
Modifiche e commenti per la routine 1	3	63
Il controllo del codice fiscale	4	29
Tecniche di ricerca	5	9
Il controllo del numero di partita IVA	5	17
I nidi For	8-9	83
BBC Acorn		
Controllo della posizione del cursore	12-13	78
C 64		
Programmer's tool kit per Commodore 64 1°. Istruzione On Error Goto nn per il VIC 20 e il C 64	12-13	24
Programmer's tool kit per Commodore 64 2°. Disabilitazione List: la nuova funzione di Append	14	80
	14	18
Come realizzare la funzione Append	14	68
Impariamo il linguaggio macchina con il VIC e il C 64 1°	14	69
	15	30
La gestione della tastiera	15	91
Sprite per C 64 1°	16	75
La gestione dei dati sul VIC e C 64	16	105
Impariamo il linguaggio macchina con il VIC e il C 64 2°	17	26
Gli "orologi" nel C 64	17	89
Il C 64 in italiano	17	43
Impariamo il linguaggio macchina con il VIC e il C 64 3°	18	57
Il C 64 e la memoria nascosta	18	101
Rivisitiamo insieme il programmer's tool kit	19	70
Impariamo il linguaggio macchina con il VIC e il C 64 4°	19	92
Magic Key	19	105
Salti etichettati per C 64	20	48
Impariamo il linguaggio macchina con il VIC e il C 64 5°	20	93
Procedura autostart	20	108
Da un Commodore all'altro	21	39
Impariamo il linguaggio macchina con il VIC e il C 64 6°	21	111
Una raffica di trucchi	21	116
Impariamo il linguaggio macchina con il VIC e il C 64 7°	22	75
Alcune scoperte sul drive 1541	22	97
I super BASIC 1°	23	70
Alta risoluzione	24	46
Come aggiungere nuovi comandi BASIC al C 64	24	73
Impariamo il linguaggio macchina con il VIC e il C 64 8°	24	82
I super BASIC 2°	24	96
Sprite Editor	25	22
Super Assembler	25	66
Type-Writer e dimensionamento dinamico	25	102
Beep e errori "sonorizzati"	26	86
Gli interrupt del chip video del C 64	27	102
Appunti di utilità e Gosub con tasti funzione	28	101
RAM test	29	12
Monitor 64	29	68
File Examiner	29	104
Easy Video 64	30	41
Uso delle label simboliche	30	96
On interval Gosub C 64	31	90
Nuova Elettronica		
Esa e ASCII	7	60
PET/CBM		
Un assembler BASIC per il PET/CBM	2	33
Effetti simultanei nei giochi di movimento	3	14
Comunicazione da PET a PET attraverso la "User Port"	3	73
Tempus fugit	5	43
Overlay	7	57
Un tasto per una istruzione	8-9	85
Un overlay più flessibile	10-11	64
MSX		
Uno sguardo nella VDP RAM	32	45
Sharp		
Tre nuove istruzioni	14	72
La mappa di memoria del PC-1251	15	78
Alla scoperta di nuovi caratteri	16	110
Alla ricerca del comando nascosto	17	84
I numeri misteriosi	19	98
Le subroutine del sistema operativo	20	112
I byte di controllo del display	21	118
La rappresentazione dei numeri	22	110
Alla scoperta del linguaggio macchina	24	100
Come sproteggere i programmi	26	88
Autostart e copia programmi	27	106
Struttura interna delle istruzioni BASIC	28	108
L'architettura del sistema	29	109
Caratteri & colori	30	102
Programmiamo in Assembler	31	93
Sinclair ZX80		
Rem come dati	1	95
La programmazione dei giochi di movimento	3	8
Tecniche di velocizzazione	4	64
Read, Data e Restore e l'uso del registratore	5	45
Conversione di programmi per ZX81 e ZX80 nuova ROM 1°	8-9	9
Conversione di programmi per ZX81 e ZX80 nuova ROM 2°	10-11	14
Conversione di programmi per ZX81 e ZX80 nuova ROM 3°	12-13	8
Sinclair ZX81		
Tecniche di velocizzazione	4	64
Read, Data e Restore e l'uso del registratore	5	45
Le variabili di sistema nello ZX81	7	15
Caratteri minuscoli su ZX printer	7	62
Conversione di programmi per ZX81 e ZX80 nuova ROM 1°	8-9	9
Scritte giganti, grafici e disegni, routine e messaggi video	8-9	90
Conversioni di programmi per ZX81 e ZX80 nuova ROM 2°	10-11	14
Area per linguaggio macchina nel BASIC ZX81	10-11	46
Risparmiamo memoria	10-11	64
Rotazione bidirezionale sul video dello ZX81	10-11	74
Conversione di programmi per ZX81 e ZX80 nuova ROM 3°	12-13	8
Linguaggio macchina per ZX81 1°	12-13	52
Linguaggio macchina per Sinclair 2°	15	8

Indice generale software

Routine in linguaggio macchina per ZX81 1° . 16 8
Ordinamenti con lo ZX81 ..... 17 32
Routine in linguaggio macchina per ZX81 2° . 18 87
Mini monitor per ZX81 ..... 18 109
Routine in linguaggio macchina 28 12

Sinclair ZX Spectrum
Le più interessanti caratteristiche
di gestione video 5 46
Guida alla conversione dallo ZX81
allo Spectrum 7 27
L'uso della tastiera nei programmi
di movimento 7 53
Come programmare uno schermo
scorrevole 8-9 96
Risparmiamo memoria 10-11 64
Tecniche di input e output per lo ZX
Spectrum 12-13 20
La funzione ScreenS: come utilizzarla
anche per caratteri grafici 12-13 88
Effetti sonori particolari con lo ZX Spectrum . 12-13 91
Alcuni trucchi dello Spectrum 14 65
Come proteggere i vostri programmi 14 70
Semplice renumber per Spectrum 15 70
Ingrandimento o riduzione di caratteri 15 72
Come incolonnare correttamente i numeri
senza ricorrere a una subroutine 15 79
Ideazione di caratteri speciali 16 18
Risparmio di memoria con insiemi
di caratteri interi 16 108
Eliminare blocchi di programmi senza
cancellare le linee singolarmente 17 87
Llist e Copy in doppia altezza 18 105
Conversione veloce decimale-binario 18 109
Controllo del video dal linguaggio macchina. 19 12
Lista delle variabili per Spectrum 19 14
Semplificazione della procedura di In
da tastiera 19 102
On Error Goto e disabilitazione del tasto
Break 20 114
Simulazione di joystick da tastiera 21 20
Routine in linguaggio macchina
per gestione video 21 120
Routine per Spectrum 22 68
End Of File: da errore a funzione 22 118
Memorizzazione di immagini 23 40
Trucchi e bug per tutti i gusti 23 84
L'uso e ... l'abuso dei caratteri di controllo 24 102
Super Converter per Spectrum 25 84
Extended BASIC 25 107
Diamo ai programmi BASIC la massima
qualità nella grafica di animazione 27 97
Grendel, il vostro renumber 28 63
Come (tentare di) recuperare programmi
BASIC 28 104
Tasti funzione per Spectrum 48 Kbyte 29 53
Routine 2 per Spectrum 30 81
Da una cassetta a microdrive 30 104
Sprite 31 44
Screen Input 31 86
La gestione del video 32 20

Texas TI99/4A
Input e stampa estesa 7 55
Print e Input in un contesto grafico 8-9 88
Operatori logici per TI99/4A 10-11 78
I file su cassetta nel TI99/4A 12-13 66
Rette ad alta risoluzione 12-13 82
Videografici 12-13 84
Come controllare la memoria del TI99/4A 12-13 92
Il movimento del TI BASIC 14 73
Il movimento in TI Extended BASIC 15 81
Print Using in TI BASIC 17 85
Grafica avanzata sul TI99/4A 18 75
Sprite Editor per TI99/4A 19 80
Uso delle tabelle con TI99/4A 19 86
Gestione file con un solo registratore 19 99
Apertura di un file nel modo Append 21 121
Accesso diretto in un file sequenziale 22 115
Sottoprogrammi in TI Extended BASIC 23 80
Ring 24 105
Recupero degli errori 25 106
Sub Copy 26 89
Palla che rimbalza 29 102
Hi-Res 30 52
Quali segreti ci riserva la stampa? 30 107

TRS-80
File comandi 1 86
Per chi ha il modello 1 ..... 3 10

VIC 20
Doppio schermo sul VIC 3 12
Pixelator, generatore di caratteri per il VIC ... 3 78
Come realizzare un listato bidirezionale 4 66
Disabilitazione del tasto Run/Stop 5 49
Come stampare lo schermo del VIC 7 39
Routine di interrogazione dei joystick per
VIC 20 ..... 7 56
Procedura di recupero 8-9 49
Gestione del cursore del VIC 20 ..... 8-9 93
Restore nn per il VIC 20 ..... 10-11 62
Istruzione On Error Goto nn per il VIC 20 e il
C 64 ..... 12-13 80
Disabilitazione List: la nuova funzione
di Append 14 68
Come realizzare la funzione di Append 14 69
Impariamo il linguaggio macchina
con il VIC e il C 64 1° 15 30
La gestione della tastiera 15 91
La gestione dei dati sul VIC e C 64 ..... 16 105
Impariamo il linguaggio macchina
con il VIC e il C 64 2° 17 26
Impariamo il linguaggio macchina
con il VIC e il C 64 3° 18 57
Impariamo il linguaggio macchina
con il VIC e il C 64 4° 19 92
Magic Key 19 105
Impariamo il linguaggio macchina
con il VIC e il C 64 5° 20 93
Procedura autostart 20 108
Da un Commodore all'altro 21 39
Impariamo il linguaggio macchina
con il VIC e il C 64 6° 21 111
Una raffica di trucchi 21 116
Impariamo il linguaggio macchina
con il VIC e il C 64 7° 22 75
Alcune scoperte sul drive 1541 ..... 22 97
I super BASIC 1° 23 70
Impariamo il linguaggio macchina
con il VIC e il C 64 8° 24 82
I super BASIC 2 ..... 24 96
One Touch per VIC 20 ..... 25 62
Type-Writer e dimensionamento dinamico 25 102
Beep e errori "sonorizzati" 26 86
Gli interrupt del chip video del C 64 ..... 27 102
Appunti di utilità e Gosub con tasti funzione.. 28 101

Giochi

Titolo	P.S n.	Pag.
Apple		
Il gioco del 15	1	75
Cannonate	2	62
Ippica	2	66
Galaxia	2	67
Collisione	3	93
Tastiera d'organo	3	98
Il gioco del 15	4	71
Generatore di labirinti per Apple II e ZX81	4	73
Schermo Apple-C 64 e la ragnatela	18	37
Giochiamo a Bridge con Apple II	22	15
Torneo di Bridge 1°	23	57
Torneo di Bridge 2°	24	60
Caccia alla coppia per Apple II, C 64, C 16	28	41
Binachi & Neri per Apple	28	55
Nuovo torneo di Bridge	29	58
Scopa a tre carte	31	22
Cartelle per tutti	33	26
Battaglia galattica	34	53
Atari		
Cuore matto	1	89
Animazione 3D	1	91
Caleidoscopio	1	92
Othello	3	100
Boing	3	102
Baseball	3	103
Comporre musica con l'Atari	5	24
Destroy the city con l'Atari	32	18
BASIC		
La torre di Hanoi	3	20
BBC Acorn		
Rally	3	96
C 16		
La battaglia del lago ghiacciato per C 16	27	69
Caccia alla coppia per Apple II, C 64, C 16	28	41
Box	31	14
C 64		
Quattro programmi per il Commodore 64	8-9	27
Labirinto di ghiaccio e Dot	12-13	14
Schermo Apple-C 64 e la ragnatela	18	37
Othello - Reversi per VIC 20 e C 64	18	81
Death per C 64	19	40
Gioco della Scopa a tre carte 1°	20	37
Città per C 64	20	85
Gioco della Scopa a tre carte 2°	21	82
Atterraggio	22	22
Awele	22	62
Space Traveller	23	18
Dentro l'avventura 1°	23	35
Frogger per C 64	23	44
Dentro l'avventura 2°	24	92
Dentro l'avventura 3°	26	34
L'ultimo perde	26	48
Gollum 1°	27	37
Dentro l'avventura 4°	27	75
Caccia alla coppia per Apple II, C 64, C 16	28	41
Gollum 2°	28	70
Skywalker per C 64	28	87
Grand Prix 1°	31	55
Music Editor	33	14
Grand Prix 2°	34	37
Casio		
Logica aristotelica con la calcolatrice tascabile	6	55
DAI		
Il giro del cavallo	8-9	64
Hewlett-Packard		
Master Mind su HP-29C	6	42
PET/CBM		
Il gioco del 15	1	75
Bersaglio	1	79
Orologio digitale	1	81
Frasi inutili	2	75
Generatore di labirinti	3	67
Gioco del calcio su PET/CBM	4	67
Il gioco del NIM	4	71
Il gioco del 15	4	71
"Wei-ch'i", un gioco cinese per PET	12-13	34
Abukir 1798, battaglia navale per CBM 1°	15	20
Abukir 1798 2°	16	82
La battaglia del lago ghiacciato 1°	17	34
La battaglia del lago ghiacciato 2°	18	8
Rally 1°	20	12
Gioco della Scopa a tre carte 1°	20	37
Rally 2°	21	48
Gioco della Scopa a tre carte 2°	21	82
Sharp		
Giochiamo d'azzardo sulle PC-1500	6	62
Giochiamo con i numeri di linea	18	104
Atterraggio	22	22
Tombola per Sharp MZ-700	23	14
Organo elettronico	25	40
Sinclair ZX80		
Odissea nello spazio	1	93
Roulette russa	1	94
Reverse	1	95
Sinclair ZX81		
Pontoon	2	82
Derby	2	84
Musica	2	84
Tira e molla	3	108
Trappola	3	109
Stemma	3	109
Pianeta ZX	3	110
Generatore di labirinti per Apple II e ZX81	4	73
Project robot: il robotwar per lo ZX81	7	21
Paroliamo	8-9	62
Giochi di inseguimento	10-11	44
Rally per ZX81	10-11	71
Dama cinese e Motocross	14	56
Frogger per Sinclair ZX81	16	31
Dama cinese per Sinclair ZX81	16	34
Giochi africani per Spectrum e ZX81	16	45
Othello per ZX81 1°	17	44
Satel per ZX81	17	54
ZX81 Croupier	18	24
Othello per ZX81 2°	18	63
Le torri di Plutone	20	99
Dama per ZX81	25	46
Aliens per ZX81	26	40
Sinclair ZX Spectrum		
Contraerea: le capacità grafiche dello ZX Spectrum	5	40
Labirinto per Spectrum	10-11	71
Asteroidi per ZX Spectrum	12-13	30
Othello per ZX Spectrum	14	53
Superman	15	54

Indice generale software

Collisioni per ZX Spectrum 15 91
Giochi africani per Spectrum e ZX81 ..... 16 45
Rally Safari 17 8
Spacman per ZX Spectrum 17 63
Robot a caccia 18 69
Nimble per Spectrum 19 20
Walter il raccoglimele 20 65
Cariche 21 44
Il Castello 25 50
Memo-Test per Spectrum 26 52
Quadrati per Spectrum 26 72
Baskit per Spectrum 27 43
Oroscopo per Spectrum 27 65
Zero e croci con lo Specrrum 28 93
Quoziente intelligenza 29 96
Cubo Magico 30 47
Geo-Race 30 77
Cartelle per tutti 33 26

Texas Instruments
Quadratini e lacune per TI58-59 ..... 6 31

Texas TI99/4A
Archivio scacchistico per TI99/4A 8-9 43
Il taglio del TI99/4A 10-11 56
Stid Poker 20 21
Atterraggio 22 22
Atterraggio pericoloso 23 64
Roulette 25 58
Bioritmi 26 54
Caccia al tredici 28 37
I Ching: il Libro dei Mutamenti 25 80

TRS-80
Il gioco del 15 ..... 1 75
Bersaglio 1 79
Colpo strategico 1 83
Riflessioni 2 69
Bombardiere 2 72
Labirinto mobile 2 74
Julia 2 77
Backgammon 3 89
Il gioco del 15 ..... 4 71

VIC 20
Star war 2 79
Crazy car 2 80
Motocross 2 81
Supercaccia 3 106
Collisioni per il VIC 7 68
Torre di Hanoi animata per il VIC 7 71
Master Mind contro il VIC 8-9 17
Dedalo 3D 10-11 54
Project-Robot 10-11 71
I boss del petrolio 1° 16 70
Missione UFO 17 80
Othello - Reversi per VIC 20 e C 64 ..... 18 81
Phantoms labirinto 25 80
Totocalcio: sviluppo colonnare condizionato. 33 52
Far West 34 32

Olivetti M20
Un buco nell'acqua 33 40

Sharp
Grafici di funzioni trigonometriche 28 50
Calcolo delle resistenze 31 39
Progettazione di filtri attivi 34 11

Sinclair ZX81
ZX81 Superscientifico 12-13 64
Calcolo letterale di polinomi 19 65

Sinclair ZX Spectrum
Calcoli relativi alla distribuzione normale 16 67
Analisi delle serie temporali 18 51
Risoluzione triangoli con lo Spectrum 21 32
Titolazione con lo Spectrum 24 10
Trasformate il vostro Spectrum in un registratore digitale 24 35

Texas Instruments
Calcoli ingegneristici con TI58-59 ..... 6 22
Moduli e fasi di funzioni di trasferimento con TI59 ..... 6 51
Semplificazione di frazioni per TI58-59 ..... 14 37

Texas TI99/4A
Diagrammi in TI BASIC 18 100

TRS-80
Interfacciamento: TRS-80 Pocket contro TRS-80 Mod. 1 ..... 6 47
La trasformazione automatica di coordinate tipografiche 7 35
Matematica aerobica con il TRS-80 ..... 8-9 7

Vari

Titolo	P.S. n.	Pag.
Apple		
Grafica tridimensionale con Apple	4	47
Grafica sempre a portata di mano	5	58
Un data base modulare per l'Apple	7	47
Grafica e joystick per Apple II	24	52
Gestione archivio fotografico con l'Apple	26	20
Sistemi ridotti Totocalcio 1°	26	64
Sistemi ridotti Totocalcio 2	27	59
Elaborazione a quattro mani	32	66
BASIC		
Giorno della settimana e fase della luna	7	33
C 16		
Super Spreadsheet	29	17
Data base	30	27
C 64		
Music Editor 1°	16	24
Music Editor 2°	17	14
Intelligenza artificiale e creatività	20	57
Stampiamo le ricevute fiscali	20	72
Prospettiva e grafica tridimensionale	32	52
Casio		
Agenda telefonica	30	24
DAI		
Un potente word processor	15	56
Indagine su alcune strane figure	19	46
Generico		
Dal BASIC al Pascal 1°	7	63
Dal BASIC al Pascal 2°	8-9	72
Dal BASIC al Pascal 3°	15	28
Dal BASIC al Pascal: funzioni di ingresso/uscita 4°	16	62
Dal BASIC al Pascal 5°	17	20
Dal BASIC al Pascal 6°	19	76
Dal BASIC al Pascal 7°	20	20
Intelligenza artificiale e creatività	20	57
PET/CBM		
L'autoprogrammazione: un primo passo verso l'intelligenza artificiale	4	61
Gestire file con il PET/CBM	8-9	39
Stampiamo le ricevute fiscali	20	72
MSX		
Musica con un MSX	29	49
Disegnatore per MSX	30	86
Progetto aeromodello per volo vincolato	31	75
Sharp		
Contabilità personale	20	52
Schedina 13 per Sharp MZ-700	21	75
Testi personalizzati per Sharp MZ-700	24	36
Catalogo nastri	26	12
Piani d'ammortamento con lo Sharp	27	85
Intestazione cassette	32	63
Sinclair ZX80		
Introduzione all'intelligenza artificiale 1°	10-11	22
Introduzione all'intelligenza artificiale 2°	15	36
Sinclair ZX81		
Minicalc per lo ZX81	5	50
ZX Manager	8-9	54
ZX data base	8-9	58
Introduzione all'intelligenza artificiale 1°	10-11	22
Piani di ammortamento mutui con lo ZX81	14	59
Introduzione all'intelligenza artificiale 2°	15	36
Arriva una nave carica di ... programmi da 1 Kbyte per ZX81	16	37
Sinclair ZX Spectrum		
Introduzione all'intelligenza artificiale 1°	10-11	22
Come salvare capra e cavoli nello ZX Spectrum	12-13	74
Introduzione all'intelligenza artificiale 2°	15	36
Loquax per Spectrum	19	16
Pluri-Calc	21	12
ZX Expert 1°	22	58
ZX Expert 2°	23	50
Archivio per Spectrum	24	66
Language Tutor per Spectrum 48 Kbyte	25	14
Ttmusic	26	44
Graphkit per Spectrum	27	79
Progetto aeromodello "U-Control"	29	34
Animazione di funzioni tridimensionali	32	12
La computer art secondo lo Spectrum	33	44
Sprite per disegnare	34	22
Texas TI99/4A		
Programmare grafici con il TI99/4A	8-9	23
Musica con TI99/4A	10-11	70
Grafici ad alta risoluzione con il TI99/4A	14	47
Ripartizione spese di riscaldamento	22	79
Word processor per TI99/4A	24	21
Text-Editor per TI99/4A	27	12
Bogey Boogie	33	44
VIC 20		
Picture	8-9	26
VIC Pilot	14	8
Il VIC ben temperato	17	76
Type Writer 1 1°	18	30
Type Writer 1 2°	19	30
Stampiamo le ricevute fiscali	20	72

Scientifici

Titolo	P.S n.	Pag.
Apple		
Probabilità e frequenza relativa	14	14
Calcolo combinatorio in multipla precisione	22	44
Statistica a più dimensioni	30	61
BASIC		
Un piccolo grande problema	1	17
Interi in precisione multipla	4	17
Package di aritmetica intera per 6502	4	33
Istogramma universale per PC 1211-TRS-80 Pocket	6	21
C 64		
Il casello di Melegnano per C 64	22	91
Metodi di Newton	26	60
Equadif per C 64	27	25
La simulazione dinamica di fenomeni continui 1°	30	16
La simulazione dinamica di fenomeni continui 2°	32	73
Hewlett-Packard		
Tre programmi di fisica e chimica con HP-67 e HP-97	6	33
MSX		
Un buco nell'acqua	33	40

Didattici

Titolo	P.S n.	Pag.
Apple		
La carta del cielo	3	83
PET/CBM		
La carta del cielo	5	54
Sinclair ZX/80		
Guida e cacciatore	8-9	5
ZX Spectrum		
Un esempio di analisi lessicografica	14	63
Studiare con lo Spectrum	20	17

1971-1986

EDP USA COMPIE 15 ANNI

MILANO
4-7 FEBBRAIO 1986

MOSTRA
E SEMINARI

EDP USA compie 15 anni. Quindici anni di mostre che hanno costituito un "ponte" fra la tecnologia informatica americana e la creatività applicativa italiana.
EDP USA 86 rappresenta il momento di incontro di queste forze al più alto livello specialistico.

I 6.000 mq. di area espositiva ospiteranno, nei quattro giorni della manifestazione i produttori americani più rappresentativi di tutti i comparti dell'informatica, una sezione Computer Graphics e una Trade Mission ufficiale di dieci membri specializzati in software proveniente dagli Stati Uniti. Nel salone dei congressi si svolgeranno tre seminari presieduti da eminenti personalità dell'informatica:

5 febbraio: "Infocenter: Strumenti e Organizzazione, una Realtà per il Futuro".
Coordinatore: Prof. Giancarlo Martella, docente di Trattamento dell'Informazione nell'Impresa presso il Politecnico di Milano.
Quota di partecipazione: Lit. 75.000

6 febbraio: "Un viaggio dall'Invenzione del Transistor alla Intelligenza Artificiale".
Coordinatore: Prof. Marco Somalvico, ordinario di Complementi di Programmazione e di Robotica Industriale dell'Istituto di Elettronica del Politecnico di Milano.
Quota di partecipazione: Lit. 150.000, colazione inclusa.

7 febbraio: "Personal Computer e CAD: nuove frontiere da esplorare".
Coordinatore: Ing. Roberto Favero, consulente di informatica.
Quota di partecipazione: Lit. 75.000.

Le quote di partecipazione dovranno pervenire a mezzo assegno intestato a USIMC entro il 27 gennaio. L'ingresso alla mostra è riservato agli operatori del settore.
Gli studenti sono ammessi nel pomeriggio dei giorni 4 e 7 febbraio previa prenotazione e pagamento di un biglietto ingresso di Lit. 3.000.
Orario di apertura della mostra: 9.00 - 18.00.

This collection is for all TI99/4A user. If you like please join at international TI99 forum present on site ATARIAGE.
Greetings from Italy. Astharot